On the complexity of computing with zero-dimensional triangular sets

Adrien Poteaux, Éric Schost

Modular composition modulo triangular sets and applications

Adrien Poteaux[†] [⋆] Éric Schost[†]
adrien.poteaux@lifl.fr eschost@uwo.ca
[†]: Computer Science Department, The University of Western Ontario, London, ON, Canada
[⋆]: LIFL, Université Lille1, UMR-CNRS 8022, France

November 21, 2012

Abstract

We generalize Kedlaya and Umans' modular composition algorithm to the multivariate case. As a main application, we give fast algorithms for many operations involving *triangular sets* (over a finite field), such as modular multiplication, inversion, or change of order. For the first time, we are able to exhibit running times for these operations that are almost linear, without any overhead exponential in the number of variables. As a further application, we show that, from the complexity viewpoint, Charlap, Coley and Robbins' approach to elliptic curve point counting can be competitive with the better known approach due to Elkies.

Keywords Triangular set, modular composition, power projection, finite fields, complexity

1 Introduction

Our purpose in this paper is to give complexity results for operations involving *triangular sets*. We start by recalling the definition.

Triangular sets. Let $\mathbb{F}$ be our base field, and let $\mathbf{Y} = Y_1, \ldots, Y_s$ be indeterminates over $\mathbb{F}$; we order them as $Y_1 < \cdots < Y_s$. A (monic) triangular set $\mathbf{T} = (T_1, \ldots, T_s)$, for the given variable ordering, is a family of polynomials in $\mathbb{F}[\mathbf{Y}]$ with the following triangular structure:

$$\mathbf{T} \left| \begin{array}{l} T_s(Y_1, \ldots, Y_s) \\ \quad \vdots \\ T_1(Y_1), \end{array} \right.$$

such that for all i, T_i is monic in Y_i, and T_i is reduced modulo $\langle T_1, \ldots, T_{i-1} \rangle$. Note that $\mathbf{T}$ is a zero-dimensional lexicographic Gröbner basis for the order $Y_1 < \cdots < Y_s$, with a triangular structure.

Such representations can be used to solve systems of equations, whereby the solution set is described by one, or several, triangular set(s) as above (or generalizations thereof, called regular chains, that are well suited to situations of positive dimensions). There exists a vast literature dedicated to algorithms with triangular sets, regular chains, and applications: without being exhaustive, we refer the reader to [25, 5, 33, 24, 40]. In this paper, we will be concerned with some basic subroutines at the heart of these algorithms: multiplication, inversion, norm computation modulo a triangular set, as well as change of order on the variables.

It is easy to show examples, involving very few variables, where these operations are useful. The following is taken from [35]: suppose that we wish to find a factor of the self-reciprocal polynomial $T_1 = Y^6 - 5Y^5 + 6Y^4 - 9Y^3 + 6Y^2 - 5Y + 1$. The set of roots of T_1 is globally invariant under the map $\alpha \mapsto \frac{1}{\alpha}$, so the function $\alpha \mapsto \alpha + \frac{1}{\alpha}$ is invariant for this action. Hence, it is natural to introduce the bivariate triangular set

$$\mathbf{T} \left| \begin{array}{l} T_2 = Y_2 - (Y_1 + \frac{1}{Y_1}) \bmod T_1 = Y_2 - (Y_1^5 - 5Y_1^4 + 6Y_1^3 - 9Y_1^2 + 5Y_1 - 5) \\ T_1(Y_1) = Y_1^6 - 5Y_1^5 + 6Y_1^4 - 9Y_1^3 + 6Y_1^2 - 5Y_1 + 1. \end{array} \right.$$

Now, change the order of Y_1, Y_2 in $\mathbf{T}$; we obtain another triangular set that generates the same ideal:

$$\left| \begin{array}{l} Y_1^2 - Y_2 Y_1 + 1 \\ Y_2^3 - 5Y_2^2 + 3Y_2 + 1. \end{array} \right.$$

We factor the last polynomial as $Y_2^3 - 5Y_2^2 + 3Y_2 + 1 = (Y_2^2 - 4Y_2 - 1)(Y_2 - 1)$, and keep for instance the factor $Y_2^2 - 4Y_2 - 1$. Then, we restore the initial order in the system. This yields

$$\left| \begin{array}{l} Y_2 + Y_1^3 - 4Y_1^2 - 4 \\ Y_1^4 - 4Y_1^3 + Y_1^2 - 4Y_1 + 1, \end{array} \right.$$

where we can read off a factor of the initial polynomial T_1. Hence, through change of order, we were able to halve the degree of the polynomial to factor. The last section of this paper will present a less trivial application of this idea to elliptic curve point counting.

Complexity issues. Despite a growing literature, the complexity of the former operations remains imperfectly understood. For instance, in the previous example, it is not clear *a priori* that the cost of change of order would not offset the gains obtained by reducing the degree in the factorization.

To measure costs, we will write $d_i = \deg(T_i, Y_i)$, and $\mathbf{d} = (d_1, \ldots, d_s)$ will be called the *multidegree* of $\mathbf{T}$. Then, $\delta_{\mathbf{d}} = d_1 \cdots d_s$ is the natural complexity measure associated to computations modulo $\langle \mathbf{T} \rangle$, as it represents the dimension of $\mathbb{F}[\mathbf{Y}]/\langle \mathbf{T} \rangle$. The objective of our work is to give algorithms with a running time linear in $\delta_{\mathbf{d}}$, up to logarithmic factors.

The simplest non-trivial question is multiplying two polynomials A, B modulo $\langle \mathbf{T} \rangle$, assuming A and B are initially reduced modulo $\langle \mathbf{T} \rangle$. As of now, there is no known algorithm with a quasi-linear cost. For instance, the modular multiplication algorithm of [31] starts by expanding the product AB, then reduces it modulo $\langle \mathbf{T} \rangle$. As a result, an overhead exponential in the dimension appears: after expansion, the product AB has $\delta' = (2d_1 - 1) \cdots (2d_s - 1)$ monomials; we always have $\delta' \leq 2^s \delta_{\mathbf{d}}$ and in the extreme case $d_1 = \cdots = d_s = 2$ we have

$\delta' = \delta_{\mathbf{d}}^{\log_2(3)}$. Presently, the best generalist algorithm is that of [31], with a cost of $4^s\delta_{\mathbf{d}}$ base field operations, up to polylogarithmic factors; see also [9] for some particular cases.

The next question is that of inversion modulo $\langle \mathbf{T} \rangle$, when possible. The best previous known result for this question [15] also has a cost of the form $K^s\delta_{\mathbf{d}}$, up to polylogarithmic terms, for some (large) constant K. It should be pointed out that this algorithm does more than inversion: it allows one to handle zero-divisors modulo $\langle \mathbf{T} \rangle$, by splitting $\mathbf{T}$ when needed.

Next, we consider *norm* computation: by analogy with the case of field extensions, the norm of an element A in $\mathbb{F}[\mathbf{Y}]/\langle \mathbf{T} \rangle$ is the determinant of the endomorphism of multiplication by A modulo $\langle \mathbf{T} \rangle$; it coincides with the *iterated resultant* $\mathrm{res}_{Y_1}(\cdots(\mathrm{res}_{Y_s}(A, T_s)\dots), T_1)$, which is used for instance in algorithms for parametric systems [45]. We do not know of a published complexity estimate for this question; the techniques of [15] could possibly be applied and yield a result of the form $K^s\delta_{\mathbf{d}}$ (up to the usual polylogarithmic factors).

The former algorithms run in quasi-linear time when s is fixed: the challenge is to remove the exponential overhead in s. For our last question, change of order, the situation is much worse. On input $\mathbf{T}$, this problem consists in finding a triangular set $\mathbf{T}'$ for a new variable order, that generates the same ideal as $\mathbf{T}$ (provided such a $\mathbf{T}'$ exists). As of now, there is no quasi-linear algorithm for this task, even when the number of variables is kept constant (actually, even for $s = 2$).

Modular composition and power projection. A main ingredient for the algorithms to follow are operations called *modular composition* and *power projection*. These operations are well-known for univariate polynomials [11, 43], in which case they respectively read as follows:

- *modular composition*: given polynomials G, H in $\mathbb{F}[Y]$, with $\deg(G) < d$ and $\deg(H) = d$, and F in $\mathbb{F}[X]$, with $\deg(F) < e$, compute $F(G) \bmod H$
- *power projection*: given polynomials G, H in $\mathbb{F}[Y]$, with $\deg(G) < d$ and $\deg(H) = d$, an $\mathbb{F}$-linear form $\tau : \mathbb{F}[Y]/H \to \mathbb{F}$, and a bound e, compute $\tau(G^i \bmod H)$ for all $i < e$.

Over an abstract field (in an algebraic complexity model), no quasi-linear algorithm is known for these operations: the most well-known results are due to Brent-Kung [11] and Shoup [43], with a cost of $O(d^{(\omega+1)/2})$ for $e = d$, where ω is a feasible exponent for matrix multiplication (here we assume $\omega > 2$, otherwise logarithmic factors appear). For the best known value of $\omega = 2.37$ [14], we get an exponent of about 1.69. Huang and Pan showed that using rectangular matrix multiplication, one can reduce the exponent to 1.67 [23].

The starting point for this work is a recent result by Kedlaya and Umans [29]: when $\mathbb{F}$ is a finite field, they came up with quasi-linear time algorithms for these problems, in a boolean RAM model (where bit operations, not field operations, are counted). They actually do more, by considering an m-uple $(G_1, \dots, G_m)$ of polynomials instead of G, and computing respectively $F(G_1, \dots, G_m) \bmod H$, for some multivariate F, or values of the form $\tau(G_1^{a_1} \cdots G_m^{a_m})$.

Part of our tasks will be to extend these results to multivariate situations. Indeed, it has been known for long that modular composition and power projection are important for algorithms involving triangular sets: this is in essence in [43] for some particular cases, and detailed in [35].

To state the multivariate versions, we need the following notation: for $\mathbf{d} = (d_1, \ldots, d_s)$ in $\mathbb{N}^s$, $\mathbb{F}[\mathbf{Y}]_{\mathbf{d}}$ denotes the $\mathbb{F}$-vector space of polynomials $F \in \mathbb{F}[\mathbf{Y}]$ with $\deg(F, Y_i) < d_i$ for all $i \leq s$. If $\mathbf{T}$ is a triangular set of multidegree $\mathbf{d}$ in $\mathbb{F}[\mathbf{Y}]$, $R_{\mathbf{T}}$ will represent the residue class ring $\mathbb{F}[\mathbf{Y}]/\langle \mathbf{T} \rangle$. Remark that $R_{\mathbf{T}} \simeq \mathbb{F}[\mathbf{Y}]_{\mathbf{d}}$ as a vector space; as a consequence, in all our algorithms, elements of $R_{\mathbf{T}}$ are represented on the monomial basis $\{Y_1^{a_1} \cdots Y_s^{a_s} \mid 0 \leq a_i < d_i \text{ for all } i\}$. Then, multivariate modular composition, with parameter $\mathbf{e} = (e_1, \ldots, e_m) \in \mathbb{N}^m$, is the following problem:

- *multivariate modular composition*: given $\mathbf{T}$, F in $\mathbb{F}[X_1, \ldots, X_m]_{\mathbf{e}}$ and $(G_1, \ldots, G_m)$ in $R_{\mathbf{T}}^m$, compute $F(G_1, \ldots, G_m) \in R_{\mathbf{T}}$.

Remark that the classical version of this question has $m = s = 1$, and Kedlaya-Umans' result has m arbitrary and $s = 1$ (under some restrictions on $\mathbf{e}$). Remark also that in the particular case $m = 1$ and $F = X_1^2$, modular composition boils down to squaring modulo $\langle \mathbf{T} \rangle$, so it is already non-trivial.

To discuss power projection, we let $R_{\mathbf{T}}^* = \mathrm{Hom}_{\mathbb{F}}(R_{\mathbf{T}}, \mathbb{F})$ be the dual of $R_{\mathbf{T}}$ over $\mathbb{F}$; naturally, the elements of $R_{\mathbf{T}}^*$ will be given on the dual basis of the monomial basis seen before. Then, the multivariate version of power projection, with parameter $\mathbf{e}$ as above, reads as follows:

- *multivariate power projection*: given $\mathbf{T}$, $(G_1, \ldots, G_m)$ in $R_{\mathbf{T}}^m$ and τ in $R_{\mathbf{T}}^*$, compute the values $\tau(G_1^{a_1} \cdots G_m^{a_m})$, for $0 \leq a_i < e_i$, $i = 1, \ldots, m$.

Modular composition is $\mathbb{F}$-linear in the coefficients of F; the transpose map is precisely power projection (this was noted in [43] in the univariate case). Indeed, the former problem amounts to multiplying the $\delta_{\mathbf{d}} \times \delta_{\mathbf{e}}$ matrix M whose columns are the coefficients of the polynomials $G_1^{a_1} \cdots G_m^{a_m} \bmod \langle \mathbf{T} \rangle$, for $0 \leq a_1 < e_1, \ldots, 0 \leq a_m < e_m$, by the $\delta_{\mathbf{e}} \times 1$ column vector of coefficients of F. Then, the dual problem amounts to multiplying the matrix M on the left by a $1 \times \delta_{\mathbf{d}}$ vector, which we see as the coefficient vector of a linear form $\tau \in R_{\mathbf{T}}^*$.

Main results. We will revisit the questions for triangular sets discussed previously, and provide new estimates, under the additional assumptions that *(i)* the base field is a finite field $\mathbb{F}_q$ and *(ii)* $\langle \mathbf{T} \rangle$ is a radical ideal, in which case we say that $\mathbf{T}$ is *squarefree*.

The following notation is in use: if S is a set and g is a real-valued function on S, $\mathrm{plog}(g)$ denotes a real-valued function h on S for which there exists $\alpha, \beta > 0$ such that $h(s) \leq \alpha \log_2(\max(g(s), 2))^\beta$ holds for all s in S (so using this notation allows us to omit big-Os). If we do not indicate otherwise, the constant α implied in a plog( ) is universal; if it does depend on some parameters (typically a parameter ε), we indicate them in subscript. The constant β will always be universal (it won't depend on any parameter such as ε).

Our algorithms crucially rely on Kedlaya and Umans' results cited previously [29]. As a consequence, the complexity results are expressed in a similar manner: typically, for any $\varepsilon > 0$, one can obtain a running time of the form $\delta_{\mathbf{d}}^{1+\varepsilon} \log(q)\, \mathrm{plog}_\varepsilon(\log q)$, with a (large) constant hidden in the term $\mathrm{plog}_\varepsilon(\log q)$. As in [29], these results are expressed in a boolean RAM model (we may e.g. use the logarithmic cost model [2]).

To be complete, we must precise how the elements of $\mathbb{F}_q$ are encoded: elements of $\mathbb{F}_p$, for p prime, are represented as integers in $\{0, \ldots, p-1\}$; for $q = p^n$, $\mathbb{F}_q$ is assumed to be given

as $\mathbb{F}_p[T]/\langle P\rangle$, with P irreducible, so elements of $\mathbb{F}_q$ are represented as polynomials over $\mathbb{F}_p$ of degree less than n. With this representation, arithmetic operations in $\mathbb{F}_q$ can be done in time $\log(q)\,\mathrm{plog}(\log(q))$ in our RAM model (disregarding the cost induced by fetching and storing data, which depends on the data location in memory).

The algorithms are Las Vegas (we give expected running time, but results are always correct), as we rely on the random selection of field elements. Thus, we assume that our RAM can produce a random integer uniformly distributed in the range $\{0,\dots,p-1\}$ in time $\log(p)\,\mathrm{plog}(\log(p))$.

Finally, for modular composition and power projection, we add the constraint that $\mathbf{e}$ be of the form $\mathbf{e}=(e_1,e_2)$, that is, we take $m=2$: this covers the most useful applications.

Theorem 1. *Fix $\varepsilon>0$. Given a triangular set $\mathbf{T}$ of multidegree $\mathbf{d}=(d_1,\dots,d_s)$ in $\mathbb{F}_q[Y_1,\dots,Y_s]$, one can do the following using an expected $s^2\,\delta_{\mathbf{d}}^{1+\varepsilon}\log(q)\,\mathrm{plog}_\varepsilon(\log q)$ bit operations:*

- *test whether $\mathbf{T}$ is squarefree,*
- *if $\mathbf{T}$ is squarefree, multiplication, invertibility test and inversion, norm computation in $R_{\mathbf{T}}$.*

With notation as above, for $\mathbf{e}=(e_1,e_2)$ in $\mathbb{N}^2$, one can do the following using an expected $s^2\,(\delta_{\mathbf{d}}+\delta_{\mathbf{e}})^{1+\varepsilon}\log(q)\,\mathrm{plog}_\varepsilon(\log q)$ bit operations:

- *if $\mathbf{T}$ is squarefree, modular composition and power projection modulo $\langle\mathbf{T}\rangle$, with parameter $\mathbf{e}$.*

We continue the presentation of our results with change of order. For this question, we will need a stronger assumption than before: the characteristic of $\mathbb{F}_q$ must be large enough.

Theorem 2. *Fix $\varepsilon>0$. Given a squarefree triangular set $\mathbf{T}$ of multidegree $\mathbf{d}=(d_1,\dots,d_s)$ in $\mathbb{F}_q[Y_1,\dots,Y_s]$, one can do the following using an expected $s^2\,\delta_{\mathbf{d}}^{1+\varepsilon}\log(q)\,\mathrm{plog}_\varepsilon(\log q)$ bit operations, provided the characteristic of $\mathbb{F}_q$ is greater than $\delta_{\mathbf{d}}$:*

- *given a target order $Y_{\sigma(1)}<\cdots<Y_{\sigma(s)}$ on the variables, determine whether the ideal $\langle\mathbf{T}\rangle$ is generated by a triangular set $\mathbf{T}'$ for this order;*
- *if so:*
  - *compute $\mathbf{T}'$;*
  - *given A in $R_{\mathbf{T}}$, compute its image in $R_{\mathbf{T}'}$;*
  - *given A in $R_{\mathbf{T}'}$, compute its image in $R_{\mathbf{T}}$.*

Comments and relation to previous work. For most items above, except modular composition and power projection, the input and output bit sizes are essentially $\delta_{\mathbf{d}}\log(q)$; for modular composition and power projection, the input and output have bit size $(\delta_{\mathbf{d}}+\delta_{\mathbf{e}})\log(q)$. Thus, our cost estimates of respectively $s^2\,\delta_{\mathbf{d}}^{1+\varepsilon}\log(q)\,\mathrm{plog}_\varepsilon(\log q)$ and $s^2\,(\delta_{\mathbf{d}}+\delta_{\mathbf{e}})^{1+\varepsilon}\log(q)\,\mathrm{plog}_\varepsilon(\log q)$ are close to *linear*.

The term s^2 is rather inconsequential. In many cases, we can make the assumption that $d_i \geq 2$ for all i. Indeed, if $d_i = 1$, T_i has the form $Y_i - r_i(Y_1, \ldots, Y_{i-1})$ so if they are not essential to the problem at hand, Y_i and T_i can be dismissed altogether. Under this assumption, s becomes logarithmic in $\delta_{\mathbf{d}}$.

For a fixed ε, recall that the constant factor in the term $\mathrm{plog}_\varepsilon(\log q)$ is fixed as well. Thus, for multiplication and inversion, our results complement former ones: in a fixed number of variables, previous results of the form $K^s \delta_{\mathbf{d}} \mathrm{plog}(\delta_{\mathbf{d}})$ operations in $\mathbb{F}$ are marginally better, as they do not involve the factor $\delta_{\mathbf{d}}^{\varepsilon}$; our results get better when s grows large with respect to $\delta_{\mathbf{d}}$, for instance when $d_i = 2$ for all i. In this case, our results are of the form $\delta_{\mathbf{d}}^{1+\varepsilon}$ (forgetting about the dependency in q), whereas no previous result did better than $\delta_{\mathbf{d}}^{\log_2(3)}$.

For modular composition and power projection, our results extend those of [29], which hold only for $s = 1$ (those results actually cover different cases for $\mathbf{e}$ than we do: they have $\mathbf{e} = (e, \ldots, e)$). Other results known for $s > 1$ are in [43] and [27], which have $m = 1$ and $s = 2$, and [35], which discusses $m = 2$ and $s = 2$. These last works are based on Brent and Kung's idea, so the best cost they can obtain has the form $\delta_{\mathbf{d}}^{(\omega+1)/2}$, for $\delta_{\mathbf{e}} \simeq \delta_{\mathbf{d}}$.

For change of order, the situation is similar: no previous algorithm achieved a quasi-linear running time, even in the simplest case $s = 2$. Some previous approaches are based on resultant and gcd computations [10], but it is unknown how to obtain a subquadratic cost in $\delta_{\mathbf{d}}$ with such techniques: on examples such as the one given at the beginning of this introduction, even using a fast resultant algorithm, known techniques (either evaluation / interpolation or direct approaches [37]) take quadratic time. The algorithms in [35] (which are limited to $s = 2$) use modular composition and power projection as we do; however, they rely on the techniques inspired by Brent and Kung's algorithm discussed above, with a cost of order $\delta_{\mathbf{d}}^{(\omega+1)/2}$.

Main ideas and practical aspects. For both Theorems 1 and 2, the idea is to introduce a primitive element modulo $\langle \mathbf{T} \rangle$ (that is, a generator of $R_{\mathbf{T}}$), which allows us to replace multivariate operations by univariate ones.

The delicate point is the conversion between the multivariate and univariate representations. The basic idea, using trace formulas, is well-known. The key problem is how to compute the required traces efficiently: this is an instance of power projection, which we will solve using Kedlaya and Umans' idea. There is a subtle point here: a direct generalization of Kedlaya and Umans' algorithm gives a cost of the form $K^s \delta_{\mathbf{d}}^{1+\varepsilon}$, for some constant K (when $\delta_{\mathbf{e}} \simeq \delta_{\mathbf{d}}$). This is already better than previous results (as this is almost linear in $\delta_{\mathbf{d}}$ for fixed s), but it turns out that one can remove the exponential overhead K^s altogether. This is done by using this algorithm only for *bivariate* triangular sets (with $s = 2$), since then a term of the form K^s becomes irrelevant, and do the conversion from multivariate to univariate representations by handling one variable after the other, using only bivariate algorithms.

All algorithms are completely explicit, but it remains a challenge to make them competitive in practice. The central issue is to obtain an efficient implementation of our multivariate versions of Kedlaya and Umans' algorithms for modular composition and power projection. Just as with their original version, the constants hidden in the complexity estimates make a direct implementation of these algorithms slower than the classical solutions based on Brent and Kung's idea for inputs of realistic size. Further work is needed to solve this issue.

We also want to point out that the "higher-level" algorithms we build on top of modular composition and power projection (such as multiplication, inversion, change of order, etc) have an interest on their own; they are simple to understand and easy to implement. All they require is a subroutine for *bivariate* modular composition and power projection. For instance, they could also be implemented on top of the algorithms for modular composition and power projection given in [35], at the cost however of a worse theoretical complexity.

Contents

2 Preliminaries

This section recalls a few known algorithmic and complexity results involving triangular sets and finite fields. These results will be used all along this paper.

Algebraic complexity and bit complexity. Our first remark concerns the two models of computation that will be used in the paper. Both are RAM models: the algebraic RAM [26] and the boolean one. We will often use implicitly the following principle: given an algorithm written for an algebraic RAM over an abstract ring R, doing T operations in R, we will deduce an algorithm in the boolean model that solves the same problem over $\mathbb{F}_q$ in time $T\log(q)\,\mathrm{plog}(T)\,\mathrm{plog}(\log(q))$; the $\mathrm{plog}(T)$ term allows us to take into account the logarithmic cost induced by fetching and storing data. This assumes that the cost of index manipulations, loop control, etc, is negligible, and that all data is stored in the first $T^{O(1)}$ memory locations; this will be the case in our examples.

The transposition principle. Let $r, s \geq 1$ and let M be an $r \times s$ matrix with entries in a field $\mathbb{F}$. The *transposition principle* [12, Theorem 13.20] states that the existence of an algebraic circuit for the matrix-vector product $b \mapsto Mb$ implies the existence of a circuit with the same size, up to $O(r+s)$, to perform the transposed matrix-vector product $c \mapsto M^t c$. We will rely on the same idea, but in an algebraic RAM model; we will not offer a general proof, but rather indicate case-by-case how to do the transposition.

Note that for boolean models, there is no such transposition result; as a consequence, extra care must be taken when discussing transposed algorithms in this context (as already pointed out in [29]).

Arithmetic modulo triangular sets. We continue by describing basic algorithms for triangular sets. Let $\mathbf{T}$ be a triangular set of multidegree $\mathbf{d} = (d_1, \ldots, d_s)$ in $\mathbb{F}[\mathbf{Y}]$. We are concerned here with the cost of multiplication and reduction modulo $\mathbf{T}$. Theorem 1 in [31] shows the following:

- (F_1) given A and B in $\mathbb{F}[\mathbf{Y}]_{\mathbf{d}}$, one can compute the product $AB \bmod \langle \mathbf{T} \rangle$ in $4^s \delta_{\mathbf{d}}\,\mathrm{plog}(\delta_{\mathbf{d}})$ operations in $\mathbb{F}$.

- (F_2) given $\mathbf{d}' = (d'_1, \ldots, d'_m)$, with $d'_i \geq d_i$ for all i, and A in $\mathbb{F}[\mathbf{Y}]_{\mathbf{d}'}$, the remainder $A \bmod \langle \mathbf{T} \rangle$ can be computed in $4^s \delta_{\mathbf{d}'}\,\mathrm{plog}(\delta_{\mathbf{d}'})$ operations in $\mathbb{F}$.

Finite field embeddings. Given a finite field $\mathbb{F}_q$, and a positive integer t, we are interested in the cost of finding an embedding $\mathbb{F}_q \to \mathbb{F}_{q'}$, with $q' = q^t$. Following our convention, we suppose that $\mathbb{F}_q$ is given as $\mathbb{F}_p[T]/\langle P \rangle$, with $\deg(P) = r$, and we will look for $\mathbb{F}_{q'}$ as $\mathbb{F}_p[T']/Q$, with $\deg(Q) = rt$. Then, the key results we will use are the following.

- (F_3) One can construct in $\sqrt{p}\,\mathrm{plog}(q')$ operations in $\mathbb{F}_p$ two polynomials Q and V in $\mathbb{F}_p[T']$ such that $\iota : \mathbb{F}_p[T]/\langle P \rangle \to \mathbb{F}_p[T']/Q$ defined by $T \mapsto V$ is an embedding $\mathbb{F}_q \to \mathbb{F}_{q'}$. Given Q and V, one can compute $\iota(x)$ for x in $\mathbb{F}_q$, and $\iota^{-1}(y)$ for y in $\iota(\mathbb{F}_q)$, in $\mathrm{plog}(q')$ operations in $\mathbb{F}_p$.

Let us justify this claim. First, we compute an irreducible polynomial Q of degree rt in $\mathbb{F}_p[T']$ using $\sqrt{p}\,\mathrm{plog}(q')$ operations in $\mathbb{F}_p$ [41]. Then, we factor Q in $\mathbb{F}_q[T']$ for a similar

cost [42]. Take a factor ψ of Q in $\mathbb{F}_q[T']$ and lift it canonically to $\mathbb{F}_p[T, T']$. We then consider the system

$$\left| \begin{array}{l} \psi(T, T') \\ P(T) \end{array} \right.$$

and change the order of the variables. Since this system generates a maximal ideal, the change of order results in a set of two equations of the form

$$\left| \begin{array}{l} T - V(T') \\ Q(T'). \end{array} \right.$$

The map $\iota : \mathbb{F}_p[T]/\langle P \rangle \to \mathbb{F}_p[T']/Q$ defined by $T \mapsto V(T')$ realizes the requested embedding $\mathbb{F}_q \to \mathbb{F}_{q'}$. The polynomial V can be computed using a number of $\mathbb{F}_p$-operations polynomial in rt, thus in $\mathrm{plog}(q')$, e.g. by plain linear algebra; once V is known, computing $\iota(x)$, for $x \in \mathbb{F}_q$, takes a similar time, by modular composition. Finally, given $y \in \mathbb{F}_{q'}$ in the range of ι, one can recover its preimage in time $\mathrm{plog}(q')$, by linear algebra again.

3 Modular composition and power projection

In this section, we give our first algorithms for multivariate modular composition and power projection; we work modulo a triangular set $\mathbf{T}$ of multidegree $\mathbf{d} \in \mathbb{N}^s$, and we take a parameter $\mathbf{e} \in \mathbb{N}^m$. The cases we will need in the further sections have $m, s \leq 2$; for convenience, the following theorem emphasizes this special case.

Theorem 3. *Fix $\varepsilon > 0$ and positive integers m, s in $\{1, 2\}$. Given a triangular set $\mathbf{T}$ in $\mathbb{F}_q[\mathbf{Y}]$ of multidegree $\mathbf{d} \in \mathbb{N}^s$, one can solve the problems of multivariate modular composition and multivariate power projection modulo $\langle \mathbf{T} \rangle$, with parameter $\mathbf{e} \in \mathbb{N}^m$, using $(\delta_{\mathbf{d}} + \delta_{\mathbf{e}})^{1+\varepsilon} \log(q)\, \mathrm{plog}_\varepsilon(\log(q))$ bit operations.*

We will actually have to prove slightly more: first, we study the case where m is arbitrary and $\mathbf{e} = (e, \ldots, e) \in \mathbb{N}^m$, then the case $m = 2$ and $\mathbf{e}$ arbitrary (the former case is needed to deal with the latter). In our notation, Kedlaya and Umans dealt with the case $s = 1$ and $\mathbf{e} = (e, \ldots, e)$.

In the complexity analysis, we will assume that s is arbitrary, and fixed: the cost estimates will actually hide factors exponential in s. This will however induce no harm later on, since, as we said above, these results will be employed with $s \leq 2$ in the next sections.

3.1 Useful facts

We start with some known results about topics such as multivariate polynomial evaluation. These results will be used in this section only.

Multivariate evaluation. We consider the problem of evaluating a multivariate polynomial at a set of points, as well as its transpose. The following is (up to a minor modification) the quasi-linear result of [29, Corollary 4.3 and Theorem 7.6]. One difference is that we write the dependency in q as $\log(q)\,\mathrm{plog}(\log(q))$ rather than $\log(q)^{1+o(1)}$; this is possible by slightly

modifying the proof given in that reference (by augmenting by 1 the value of a parameter t used in the proof). The other difference is that we fix the number of variables m (the original statement had the condition $m = e^{o(1)}$), which allows us to dispense with the original condition that e be large enough.

The result we quote holds in a boolean model, so we take $\mathbb{F}_q$ as a base field. Given $\mathbf{e}$ in $\mathbb{N}^m$ and a set $B \subset \mathbb{F}_q^m$ of cardinality N, we define

$$\begin{array}{rccc} \mathsf{Eval}_B: & \mathbb{F}_q[\mathbf{X}]_{\mathbf{e}} & \to & \mathbb{F}_q^N \\ & F & \mapsto & [F(b) \mid b \in B]; \end{array}$$

the transpose map is $\mathsf{Eval}_B^t : \mathbb{F}_q^N \to \mathbb{F}_q[\mathbf{X}]_{\mathbf{e}}^*$.

(F_4) Fix $\varepsilon > 0$ and a positive integer m. Given $\mathbf{e} \in \mathbb{N}^m$ of the form $\mathbf{e} = (e, \ldots, e)$, a set $B \subset \mathbb{F}_q^m$ of cardinality N and $F \in \mathbb{F}_q[\mathbf{X}]_{\mathbf{e}}$, one can compute $\mathsf{Eval}_B(F)$ in $(\delta_{\mathbf{e}} + N)^{1+\varepsilon} \log(q)\, \mathrm{plog}_{\varepsilon,m}(\log(q))$ bit operations.

(F_5) Fix $\varepsilon > 0$ and a positive integer m. Given $\mathbf{e} \in \mathbb{N}^m$ of the form $\mathbf{e} = (e, \ldots, e)$, a set $B \subset \mathbb{F}_q^m$ of cardinality N and $u \in \mathbb{F}_q^N$, one can compute $\mathsf{Eval}_B^t(u)$ in $(\delta_{\mathbf{e}} + N)^{1+\varepsilon} \log(q)\, \mathrm{plog}_{\varepsilon,m}(\log(q))$ bit operations.

Fact F_5 is from [29, Theorem 7.6]. That reference has the extra assumption that $N = \delta_{\mathbf{e}}$; we briefly discuss how to lift this assumption. If $N \geq \delta_{\mathbf{e}}$, the input of Eval_B^t has larger cardinality than the output. Then, we do as in [29, Theorem 7.7], by solving $\lceil N/\delta_{\mathbf{e}} \rceil$ instances of size $\delta_{\mathbf{e}}$ and adding the results. If $N \leq \delta_{\mathbf{e}}$, we do not have enough points, so we add $\delta_{\mathbf{e}} - N$ dummy points and pad the input vector with zeros. In both cases, the cost fits into our claimed bound.

Structured evaluation and interpolation. Next, we discuss multivariate evaluation and interpolation at special sets of points. The following results hold over an abstract field $\mathbb{F}$. Let $\mathbf{e} = (e_1, \ldots, e_m)$ be in $\mathbb{N}^m$, and consider a subset of $\mathbb{F}^m$ of the form $B = B_1 \times \cdots \times B_m$, with B_i of cardinality e_i (thus, B is an m-dimensional grid). For input polynomials with support in $\mathbb{F}[\mathbf{X}]_{\mathbf{e}}$, evaluation and interpolation at such a grid are simple problems.

(F_6) given $F \in \mathbb{F}[\mathbf{X}]_{\mathbf{e}}$ and B as above, one can compute $\mathsf{Eval}_B(F)$ in $\delta_{\mathbf{e}} \,\mathrm{plog}(\delta_{\mathbf{e}})$ operations in $\mathbb{F}$.

(F_7) given values $v = [v_b \mid b \in B]$ and B as above, there exists a unique polynomial $F \in \mathbb{F}[\mathbf{X}]_{\mathbf{e}}$ such that $\mathsf{Eval}_B(F) = v$; one can compute F in $\delta_{\mathbf{e}} \,\mathrm{plog}(\delta_{\mathbf{e}})$ operations in $\mathbb{F}$.

The multivariate algorithms simply consists in applying the classical univariate algorithms, variable by variable; see for instance [34].

Reformating a polynomial. One of the main ideas used in [29], and before it in Umans' algorithm [44], is to simultaneously increase the number of variables and decrease the degrees of a polynomial. Our definition slightly extends the one used there, by allowing arbitrary

partial degrees. In what follows, as in the previous paragraph, our polynomials will have coefficients in an abstract field $\mathbb{F}$.

Given $\mathbf{e} = (e_1, \ldots, e_m) \in \mathbb{N}^m$, we will be interested in mapping polynomials in $\mathbb{F}[X_1, \ldots, X_m]_{\mathbf{e}}$ to polynomials in more variables, with lower degree. Let $(\ell_1, \ldots, \ell_m)$ be positive integers; to each variable X_i we will associate ℓ_i new variables $X_{i,0}, \ldots, X_{i,\ell_i-1}$, so that the total number of new variables is $m' = \ell_1 + \cdots + \ell_m$.

Consider a vector $\mathbf{e}' = (e'_1, \ldots, e'_1, \ldots, e'_m, \ldots, e'_m) \in \mathbb{N}^{m'}$, such that each e'_i is repeated ℓ_i times. This will be our new degree vector, so that we put the constraint $e_i'^{\ell_i} \geq e_i$. Then, we can define the $\mathbb{F}$-linear map $\Lambda_{\mathbf{e},\mathbf{e}'}$ by

$$\begin{array}{rccl} \Lambda_{\mathbf{e},\mathbf{e}'}: & \mathbb{F}[X_1, \ldots, X_m]_{\mathbf{e}} & \to & \mathbb{F}[X_{1,0}, \ldots, X_{m,\ell_m-1}]_{\mathbf{e}'} \\ & X_1^{a_1} \cdots X_m^{a_m} & \mapsto & X_{1,0}^{a_{1,0}} \cdots X_{1,\ell_1-1}^{a_{1,\ell_1-1}} \cdots X_{m,0}^{a_{m,0}} \cdots X_{m,\ell_m-1}^{a_{m,\ell_m-1}}, \end{array}$$

where $a_{i,0}, \ldots, a_{i,\ell-1}$ are the coefficients of the expansion of a_i in base e'_i. Next, given an $\mathbb{F}$-algebra R, we define the map $\Lambda^\star_{\mathbf{e},\mathbf{e}'}$ as

$$\begin{array}{rccl} \Lambda^\star_{\mathbf{e},\mathbf{e}'}: & R^m & \to & R^{m'} \\ & G = (G_1, \ldots, G_m) & \mapsto & (G_i, G_i^{e'_i}, \ldots, G_i^{e_i'^{\ell_i-1}})_{i=1,\ldots,m}. \end{array}$$

The key equality is then the following: for F in $\mathbb{F}[X_1, \ldots, X_m]_{\mathbf{e}}$ and G in R^m, we have $F(G) = \Lambda_{\mathbf{e},\mathbf{e}'}(F)\big(\Lambda^\star_{\mathbf{e},\mathbf{e}'}(G)\big)$. Computing $\Lambda_{\mathbf{e},\mathbf{e}'}(F)$ and $\Lambda^\star_{\mathbf{e},\mathbf{e}'}(G)$ induces a cost, which we summarize here:

(F_8) Given F in $\mathbb{F}[X_1, \ldots, X_m]_{\mathbf{e}}$, one can compute $\Lambda_{\mathbf{e},\mathbf{e}'}(F)$ in $O(\delta_{\mathbf{e}'})$ operations in $\mathbb{F}$. Given G in R^m, one can compute $\Lambda^\star_{\mathbf{e},\mathbf{e}'}(G)$ using $O(\log(\delta_{\mathbf{e}'}))$ multiplications in R.

The first point is obvious, as we simply fill an array of size $\delta_{\mathbf{e}'}$. For the second point, for a fixed $i \leq m$, we must compute $G_i, G_i^{e'_i}, \ldots, G_i^{e_i'^{\ell_i-1}}$. This is done using ℓ_i exponentiations by e'_i, that is, $O(\ell_i \log(e'_i))$ multiplications in R. The total is thus $O(\log({e'_1}^{\ell_1} \cdots {e'_m}^{\ell_m})) = O(\log(\delta_{\mathbf{e}'}))$.

3.2 The case $\mathbf{e} = (e, \ldots, e)$

We can now turn to modular composition and power projection, starting with the case where $\mathbf{e} = (e, \ldots, e)$. This situation is very close to [29, Theorem 3.1], as the only (conceptually trivial) difference is that we work modulo a triangular set, instead of a single polynomial. The proof we give follows the one given in that reference: the key idea developed in [29], and previously in [44], is to reduce the problem to multipoint evaluation.

In the following theorem, s and m are fixed, so the cost estimate hides the dependency in these parameters. The dependency in m could easily be controlled, by requiring $m = e^{o(1)}$, as in [29, Theorem 3.1]. With respect to s, however, the cost would turn out to involve a factor of the form 4^s, due to the application of facts F_1 and F_2; as said before, this is not harmful since we will use this result with $s \leq 2$.

Theorem 4. *Fix $\varepsilon > 0$ and positive integers m, s. Given a triangular set $\mathbf{T}$ in $\mathbb{F}_q[\mathbf{Y}]$ of multidegree $\mathbf{d} \in \mathbb{N}^s$, one can solve the problem of multivariate modular composition modulo $\langle \mathbf{T} \rangle$, with parameter $\mathbf{e} = (e, \ldots, e) \in \mathbb{N}^m$, using $(\delta_{\mathbf{d}} + \delta_{\mathbf{e}})^{1+\varepsilon} \log(q)\, \mathrm{plog}_{\varepsilon,s,m}(\log(q))$ bit operations.*

Proof. Without loss of generality, we may assume that $\varepsilon \leq 1$. Given a triangular set $\mathbf{T} \in \mathbb{F}_q[Y_1, \ldots, Y_s]$ of multidegree $\mathbf{d} = (d_1, \ldots, d_s)$, $(G_1, \ldots, G_m)$ in $R_{\mathbf{T}}^m$ and F in $\mathbb{F}_q[X_1, \ldots, X_m]_{\mathbf{e}}$, we will show how to compute $F(G_1, \ldots, G_m) \in R_{\mathbf{T}}$. The algorithm follows that of [29, Theorem 3.1], up to handling reduction modulo multivariate polynomials. In all that follows, remember that we have fixed ε, s, m, so they should be seen as constants.

The idea is to proceed by evaluation and interpolation. To enable this, we will replace $(m, \mathbf{d}, \mathbf{e}, q)$ by better suited parameters $(m', \mathbf{d}', \mathbf{e}', q')$. First, we define $\ell = \lceil 2s/(m\varepsilon) \rceil$, $m' = \ell m$ and $e' = \lceil e^{1/\ell} \rceil$. Remark that ℓ and m' are bounded from above by a constant. On the other hand, we have the lower bound $m' \geq 2s/\varepsilon$: m' will our new number of variables; it is large enough, but not too large.

Let next $\mathbf{e}'$ be the vector $(e', \ldots, e')$ of length m'. Finally, let $d = \max(d_1, \ldots, d_s)$, and define $\mathbf{d}' = (d'_1, \ldots, d'_s)$, with $d'_i = m'e'd_i$. Before going further, we establish the following inequalities:

- There exists a constant c_1 depending on (ε, m, s) such that $\delta_{\mathbf{e}'} \leq c_1 \delta_{\mathbf{e}}^{1+\varepsilon}$. Indeed, we have $\delta_{\mathbf{e}'} = \lceil e^{1/\ell} \rceil^{\ell m}$. We deduce the inequalities

$$\delta_{\mathbf{e}'} \leq (e^{1/\ell} + 1)^{\ell m} \quad \text{and thus} \quad \delta_{\mathbf{e}'} \leq \delta_{\mathbf{e}} (1 + e^{-1/\ell})^{\ell m}.$$

  There exists c_0 depending on (ε, m, s) such that $(1 + e^{-1/\ell})^{\ell}$ admits the upper bound $c_0 e^{\varepsilon}$ for all e, and the conclusion follows by raising to the power m and taking $c_1 = c_0^m$.

- For $\varepsilon \leq 1$, there exists a constant c_2 depending on (ε, m, s) such that $\delta_{\mathbf{d}'} \leq c_2 \delta_{\mathbf{e}}^{\varepsilon} \delta_{\mathbf{d}}$. Indeed, we have $\delta_{\mathbf{d}'} = (m'e')^s \delta_{\mathbf{d}}$. The equality $e' = \delta_{\mathbf{e}'}^{1/m'}$ implies

$$m'e' = m' \delta_{\mathbf{e}'}^{1/m'}, \quad \text{so that} \quad (m'e')^s = m'^s \delta_{\mathbf{e}'}^{s/m'}.$$

  Recall that $m' \geq 2s/\varepsilon$; then, the former equality gives $(m'e')^s \leq m'^s \delta_{\mathbf{e}'}^{\varepsilon/2}$. The upper bounds $\delta_{\mathbf{e}'} \leq c_1 \delta_{\mathbf{e}}^{1+\varepsilon} \leq c_1 \delta_{\mathbf{e}}^2$ enable us to conclude, by taking $c_2 = m'^s c_1^{\varepsilon/2}$.

We will need to ensure that the base field contains at least $m'e'd$ elements. The final correction we do is thus to change q into q', defined below; in what follows, in any case, our base field will be $\mathbb{F}_{q'}$.

- If $q \geq m'e'd$, we do nothing and we let $q' = q$.

- Else, we find an irreducible polynomial of degree $n = \lceil \log_q(m'e'd) \rceil$ over $\mathbb{F}_q$ and an embedding $\iota : \mathbb{F}_q \to \mathbb{F}_{q'}$, with now $q' = q^n$. By fact F_3, this can be done in $\sqrt{p}\,\mathrm{plog}(q')$ operations in $\mathbb{F}_p$. Remark that $q' \leq qm'e'd \leq (m'e'd)^2$, so that a quantity polylogarithmic in q' is polylogarithmic in $m'e'd$, and thus in $\delta_{\mathbf{d}} + \delta_{\mathbf{e}}$. Since $p \leq q$, and $q \leq m'e'd$, $\sqrt{p}$ is $O(\sqrt{\delta_{\mathbf{d}} \delta_{\mathbf{e}}})$, with a constant depending on ε, s, m, so the time for building $\mathbb{F}_{q'}$ is $(\delta_{\mathbf{d}} + \delta_{\mathbf{e}})\,\mathrm{plog}_{\varepsilon,s,m}(\delta_{\mathbf{d}} + \delta_{\mathbf{e}})$ operations in $\mathbb{F}_p$.

  Fact F_3 also shows that applying and inverting ι on its image, can be done in $\mathrm{plog}(q')$ operations in $\mathbb{F}_p$. In view of what was said before, this is $\mathrm{plog}_{\varepsilon,s,m}(\delta_{\mathbf{d}} + \delta_{\mathbf{e}})$ operations. As a consequence, the sum of all costs related to ι and ι^{-1} will as well be $(\delta_{\mathbf{d}} + \delta_{\mathbf{e}})\,\mathrm{plog}_{\varepsilon,s,m}(\delta_{\mathbf{d}} + \delta_{\mathbf{e}})$ operations in $\mathbb{F}_p$.

We can now explain the algorithm. To compute $F(G_1, \ldots, G_m) \bmod \langle \mathbf{T} \rangle$, we will actually compute $F'(G'_1, \ldots, G'_{m'}) \bmod \langle \mathbf{T} \rangle$, with

$$F' = \Lambda_{\mathbf{e},\mathbf{e}'}(F) \quad \text{and} \quad (G'_1, \ldots, G'_{m'}) = \Lambda^{\star}_{\mathbf{e},\mathbf{e}'}(G_1, \ldots, G_m) \bmod \langle \mathbf{T} \rangle.$$

We saw (Section 3.1, fact F_8) that computing F' and $(G'_1, \ldots, G'_{m'})$ takes $O(\delta_{\mathbf{e}'})$ operations in $\mathbb{F}_{q'}$ and $O(\log(\delta_{\mathbf{e}'}))$ multiplications modulo $\langle \mathbf{T} \rangle$. Fact F_1 shows that the cost of one multiplication modulo $\langle \mathbf{T} \rangle$ is $4^s \delta_{\mathbf{d}} \operatorname{plog}(\delta_{\mathbf{d}})$ operations in $\mathbb{F}_{q'}$, so the total is $(4^s \delta_{\mathbf{d}} + \delta_{\mathbf{e}'}) \operatorname{plog}(\delta_{\mathbf{d}} + \delta_{\mathbf{e}'})$ operations in $\mathbb{F}_{q'}$, which we may rewrite as $(\delta_{\mathbf{d}} + \delta_{\mathbf{e}'}) \operatorname{plog}_s(\delta_{\mathbf{d}} + \delta_{\mathbf{e}'})$.

To compute $F'(G'_1, \ldots, G'_{m'}) \bmod \langle \mathbf{T} \rangle$, we will first compute $\varphi = F'(G'_1, \ldots, G'_{m'})$, then reduce it modulo $\langle \mathbf{T} \rangle$. The reduction will raise no difficulty; the delicate step is the computation of φ.

This will be done by evaluation and interpolation. Remark that φ lies in $\mathbb{F}_{q'}[\mathbf{Y}]_{\mathbf{d}'}$. Thus, we choose subsets $B_1, \ldots, B_s$ of $\mathbb{F}_{q'}$ of cardinalities $d'_1, \ldots, d'_s$; this is possible by assumption on q'. We first compute all values $g'_b = (G'_1(b), \ldots, G'_{m'}(b)) \in \mathbb{F}_{q'}^{m'}$ for $b \in B_1 \times \cdots \times B_s$, then all values $f'_b = F'(g'_b)$; we finally compute φ by interpolating the values f'_b at $B_1 \times \cdots \times B_s$.

Let us postpone the cost of the evaluation of F' at the points g'_b, and estimate all other costs first. To compute all g'_b, we evaluate each G'_i at $B_1 \times \cdots \times B_s$, for $i \leq m'$. By fact F_6, each evaluation takes $\delta_{\mathbf{d}'} \operatorname{plog}(\delta_{\mathbf{d}'})$ operations in $\mathbb{F}_{q'}$, for a total of $m' \delta_{\mathbf{d}'} \operatorname{plog}(\delta_{\mathbf{d}'})$ operations in $\mathbb{F}_{q'}$. Since m' is bounded by a constant, this is $\delta_{\mathbf{d}'} \operatorname{plog}_{\varepsilon,s,m}(\delta_{\mathbf{d}'})$. By fact F_7, this also controls the cost of interpolation. Finally, since s is constant, Fact F_2 implies a cost of $4^s \delta_{\mathbf{d}'} \operatorname{plog}(\delta_{\mathbf{d}'}) = \delta_{\mathbf{d}'} \operatorname{plog}_s(\delta_{\mathbf{d}'})$ operations in $\mathbb{F}_{q'}$ for the reduction of φ modulo $\langle \mathbf{T} \rangle$.

The total cost for all previous steps is bounded from above by $(\delta_{\mathbf{d}'} + \delta_{\mathbf{e}'}) \operatorname{plog}_{\varepsilon,s,m}(\delta_{\mathbf{d}'} + \delta_{\mathbf{e}'})$ operations in $\mathbb{F}_{q'}$.

We finish by estimating the cost of computing all f'_b. Since m' is bounded by a constant, we can apply fact F_4 with parameters $\mathbf{e}'$ and $N = \delta_{\mathbf{d}'}$, to get a cost of $(\delta_{\mathbf{d}'} + \delta_{\mathbf{e}'})^{1+\varepsilon} \log(q') \operatorname{plog}_{\varepsilon,s,m}(\log(q'))$ bit operations. In view of the claim of the previous paragraph, the total time fits into this bound as well. Using the bounds given previously on $\delta_{\mathbf{e}'}$ and $\delta_{\mathbf{d}'}$, and a quick simplification, this becomes $(\delta_{\mathbf{d}} + \delta_{\mathbf{e}})^{1+3\varepsilon} \log(q') \operatorname{plog}_{\varepsilon,s,m}(\log(q'))$ for $\varepsilon \leq 1$.

Remember that $q' \leq q m' e' d$, so that $\log(q')$ is at most $\log(q) + \operatorname{plog}_{\varepsilon,s,m}(\delta_{\mathbf{e}} \delta_{\mathbf{d}})$. The polylogarithmic terms in $\delta_{\mathbf{e}} \delta_{\mathbf{d}}$ admit as well an upper bound of the form $c(\varepsilon, m, s)(\delta_{\mathbf{d}} + \delta_{\mathbf{e}})^{\varepsilon}$, and the conclusion follows, up to replacing ε by $\varepsilon/4$. □

We continue with a description of the transposition of this algorithm, that deals with power projection. The reasoning follows the one of [29, section 7.2].

Theorem 5. *Fix $\varepsilon > 0$ and positive integers m, s. Given a triangular set $\mathbf{T}$ in $\mathbb{F}_q[\mathbf{Y}]$ of multidegree $\mathbf{d} \in \mathbb{N}^s$, one can solve the problem of multivariate power projection modulo $\langle \mathbf{T} \rangle$, with parameter $\mathbf{e} = (e, \ldots, e) \in \mathbb{N}^m$, using $(\delta_{\mathbf{d}} + \delta_{\mathbf{e}})^{1+\varepsilon} \log(q) \operatorname{plog}_{\varepsilon,s,m}(\log(q))$ bit operations.*

Proof. We will show how to transpose the algorithm given in the proof of the previous theorem. Seen as a linear map in F, the former algorithm replaces F by $F' = \Lambda_{\mathbf{e},\mathbf{e}'}(F)$, performs a multipoint evaluation of F', then a multivariate interpolation at a grid, and finally a reduction modulo $\langle \mathbf{T} \rangle$.

We explain here how to transpose these four steps in reverse order (the other steps, which are non-linear, are unchanged). The last step is a modular reduction. Its transpose is described in [8] in the case $s = 1$ and in [35] for $s = 2$; in general, it suffices to transpose step-by-step the reduction algorithm of [31], and the cost remains unchanged.

The third step is a multivariate interpolation at a grid of dimension s, which is done by interpolating one variable after the other. The transposed algorithm thus requires to perform s transposed univariate interpolations; we refer to [28, 8] for such an algorithm. Again, the cost remains unchanged. The second step is a multidimensional multipoint evaluation, with monomial support $\mathbf{e}'$, at $N = \delta_{\mathbf{d}'}$ points; its transpose is handled by invoking fact F_5 (instead of fact F_4 for the forward direction). Finally, the first step is an injection, whose transpose is a projection, and takes linear time.

The costs of all transposed steps are thus the same as the ones for the forward direction, and as a consequence, the overall running time admits the same bound. □

3.3 The case $\mathbf{e} = (e_1, e_2)$

The results of the previous subsection assume that $\mathbf{e} \in \mathbb{N}^m$ has the special form $(e, \ldots, e)$. What we will actually need in the sequel are the cases $m = 1$ (which is thus covered) and $m = 2$, but in this case with $\mathbf{e} = (e_1, e_2)$ arbitrary. This subsection shows how to handle this case using the former theorems. Again, the number of variables s in our triangular set is fixed.

Theorem 6. *Fix $\varepsilon > 0$ and a positive integer s. Given a triangular set $\mathbf{T}$ in $\mathbb{F}_q[\mathbf{Y}]$ of multidegree $\mathbf{d} \in \mathbb{N}^s$, one can solve the problem of multivariate modular composition modulo $\langle \mathbf{T} \rangle$, with parameter $\mathbf{e} = (e_1, e_2) \in \mathbb{N}^2$, using $(\delta_{\mathbf{d}} + \delta_{\mathbf{e}})^{1+\varepsilon} \log(q)\, \mathrm{plog}_{\varepsilon,s}(\log(q))$ bit operations.*

Proof. Given a triangular set $\mathbf{T} \in \mathbb{F}_q[Y_1, \ldots, Y_s]$ of multidegree $\mathbf{d}$, (G_1, G_2) in $R_{\mathbf{T}}$ and F in $\mathbb{F}_q[X_1, X_2]_{(e_1,e_2)}$, we will show how to compute $F(G_1, G_2) \in R_{\mathbf{T}}$. Since the order of the variables X_1 and X_2 is irrelevant, we may assume that $e_1 \leq e_2$. We will distinguish two cases, depending on whether $e_2 \leq e_1^{1/\varepsilon}$ or not.

Suppose first that $e_2 \leq e_1^{1/\varepsilon}$ holds. Let

$$\ell_1 = \left\lceil \frac{1}{\varepsilon} \right\rceil, \quad \ell_2 = \left\lceil \frac{1}{\varepsilon} \log_{e_1}(e_2) \right\rceil \quad \text{and} \quad e = \lceil e_1^{\varepsilon} \rceil;$$

as a consequence of our assumption on e_1, e_2, both ℓ_1 and ℓ_2 are bounded by constants (since ε is fixed). Define the vector $\mathbf{e}' = (e, \ldots, e)$ in $\mathbb{N}^{\ell_1+\ell_2}$, and let further

$$F' = \Lambda_{\mathbf{e},\mathbf{e}'}(F) \quad \text{and} \quad (G'_{1,1}, \ldots, G'_{1,\ell_1}, G'_{2,1}, \ldots, G'_{2,\ell_2}) = \Lambda^{\star}_{\mathbf{e},\mathbf{e}'}(G_1, G_2) \bmod \langle \mathbf{T} \rangle,$$

so that we have

$$F(G_1, G_2) \bmod \langle \mathbf{T} \rangle = F'(G'_{1,1}, \ldots, G'_{1,\ell_1}, G'_{2,1}, \ldots, G'_{2,\ell_2}) \bmod \langle \mathbf{T} \rangle.$$

We saw in fact F_8 that F' and all $G'_{i,j}$ can be computed in $O(\delta_{\mathbf{e}'})$ operations in $\mathbb{F}_q$ and $O(\log(\delta_{\mathbf{e}'}))$ multiplications modulo $\langle \mathbf{T} \rangle$. This will be negligible compared to what follows.

Knowing the $G'_{i,j}$, we are left with an instance of modular composition modulo $\langle \mathbf{T} \rangle$ with parameter $\mathbf{e}'$. Because $\ell_1 + \ell_2$ is bounded by a constant, we can apply Theorem 4, giving a running time of $(\delta_{\mathbf{d}} + \delta_{\mathbf{e}'})^{1+\varepsilon} \log(q) \operatorname{plog}_{\varepsilon,s}(\log(q))$ bit operations. Next, using all equalities written before, we obtain the upper bound

$$\delta_{\mathbf{e}'} = e^{\ell_1+\ell_2} \leq (2\, e_1^{\varepsilon})^{\frac{1}{\varepsilon}+\frac{1}{\varepsilon}\log_{e_1}(e_2)+2} \leq 2^{\frac{1}{\varepsilon}+\frac{1}{\varepsilon^2}+2} \delta_{\mathbf{e}}^{1+\varepsilon}$$

using the upper bound $e \leq 2\, e_1^{\varepsilon}$ and, for the exponents, $\lceil x \rceil \leq x + 1$. Thus, $(\delta_{\mathbf{d}} + \delta_{\mathbf{e}'})^{1+\varepsilon}$ admits the upper bound $2^{\frac{1}{\varepsilon}+\frac{1}{\varepsilon^2}+2}(\delta_{\mathbf{d}} + \delta_{\mathbf{e}})^{1+3\varepsilon}$ for $\varepsilon \leq 1$. This finishes the proof in this case (up to replacing ε by say $\varepsilon/3$).

Next, we consider the case $e_2 \geq e_1^{1/\varepsilon}$; in particular, we have $e_1 \leq \delta_{\mathbf{e}}^{\varepsilon}$. Write $F(X_1, X_2) = \sum_{i=0}^{e_1-1} F_i(X_2) X_1^i$, with $\deg(F_i) < e_2$ for all i, and recall that we want to compute

$$F(G_1, G_2) \bmod \langle \mathbf{T} \rangle = \sum_{i=0}^{e_1-1} F_i(G_2) G_1^i \bmod \langle \mathbf{T} \rangle.$$

We proceed as follows:

1. We first compute $F_i(G_2) \bmod \langle \mathbf{T} \rangle$, for $0 \leq i \leq e_1 - 1$. Each of these computations is an instance of modular composition modulo $\langle \mathbf{T} \rangle$ with parameter $(e_2) \in \mathbb{N}^1$, that is, with $m = 1$. By Theorem 4, the cost of this step is $e_1(\delta_{\mathbf{d}} + e_2)^{1+\varepsilon} \log(q) \operatorname{plog}_{\varepsilon,s}(\log(q))$ bit operations. Since $e_1 \leq \delta_{\mathbf{e}}^{\varepsilon}$, this is at most $(\delta_{\mathbf{d}} + \delta_{\mathbf{e}})^{1+2\varepsilon} \log(q) \operatorname{plog}_{\varepsilon,s}(\log(q))$.

2. Then, we use these values in a Horner scheme to get the result in e_1 multiplications and additions in $R_{\mathbf{T}}$; this gives us a cost of $e_1 \delta_{\mathbf{d}} \operatorname{plog}_s(\delta_{\mathbf{d}})$ operations in $\mathbb{F}_q$. Using again the bound $e_1 \leq \delta_{\mathbf{e}}^{\varepsilon}$, this is $(\delta_{\mathbf{d}} + \delta_{\mathbf{e}})^{1+\varepsilon} \operatorname{plog}(\delta_{\mathbf{d}})$ operations in $\mathbb{F}_q$, and thus $(\delta_{\mathbf{d}} + \delta_{\mathbf{e}})^{1+\varepsilon} \operatorname{plog}_s(\delta_{\mathbf{d}} + \delta_{\mathbf{e}}) \log(q) \operatorname{plog}(\log(q))$ bit operations. The latter cost admits the upper bound $(\delta_{\mathbf{d}} + \delta_{\mathbf{e}})^{1+2\varepsilon} \log(q) \operatorname{plog}_{\varepsilon,s}(\log(q))$.

Replacing ε by $\varepsilon/2$ concludes the proof. □

We conclude this section with the transposed version of the former algorithm.

Theorem 7. *Fix $\varepsilon > 0$ and a positive integer s. Given a triangular set $\mathbf{T}$ in $\mathbb{F}_q[\mathbf{Y}]$ of multidegree $\mathbf{d} \in \mathbb{N}^s$, one can solve the problem of multivariate power projection modulo $\langle \mathbf{T} \rangle$, with parameter $\mathbf{e} = (e_1, e_2) \in \mathbb{N}^2$, using $(\delta_{\mathbf{d}} + \delta_{\mathbf{e}})^{1+\varepsilon} \log(q) \operatorname{plog}_{\varepsilon,s}(\log(q))$ bit operations.*

Proof. As in the proof of the previous theorem, we assume that $e_1 \leq e_2$ and we consider the two cases $e_2 \leq e_1^{1/\varepsilon}$ or $e_2 \geq e_1^{1/\varepsilon}$. In the forward direction, both cases involve modular composition (which was handled using Theorem 4), so the transpose which rely on power projection.

- In the first case, the linear part of the algorithm amounts to replacing F by F' and solving an instance of modular composition modulo $\langle \mathbf{T} \rangle$, with parameter $\mathbf{e}'$; we use Theorem 5 to do the transposed operation, power projection, in the same amount of time as in the forward direction.

- In the second case, the first step consists in solving e_1 instances of modular composition modulo $\langle \mathbf{T} \rangle$, with parameter (e_2); their transposes are all handled by Theorem 5. The second step is simply Horner's rule modulo $\langle \mathbf{T} \rangle$, and can be transposed without difficulty (see e.g. [7]).

In both cases, the costs of all transposed steps are the same as the ones for the forward direction, so the overall running time admits the same bound. □

4 Representations of zero-dimensional ideals

In this section, we change our focus: we discuss representations of zero-dimensional algebraic sets using either univariate polynomials, triangular sets, or an intermediate data structure.

For our discussion, we consider a zero-dimensional ideal I in $\mathbb{F}[\mathbf{Y}] = \mathbb{F}[Y_1, \ldots, Y_s]$, where $\mathbb{F}$ is a perfect field; we do not necessarily assume that I is defined by a triangular set for any order. Finally, we let $R = \mathbb{F}[\mathbf{Y}]/I$ be the residue class ring modulo I, and let δ be the dimension of the $\mathbb{F}$-vector space R.

To $A \in R$, we associate the multiplication-by-A endomorphisms of R, written M_A. The *minimal polynomial* and the *characteristic polynomial* of A, respectively written $m_A \in \mathbb{F}[Y]$ and $\chi_A \in \mathbb{F}[Y]$, are then defined as those of M_A. Let V be the zero-set of I in $\overline{\mathbb{F}}^s$, where $\overline{\mathbb{F}}$ is an algebraic closure of $\mathbb{F}$. Then, when I is radical, because of our perfectness assumption, we have the factorization (over $\overline{\mathbb{F}}$)

$$\chi_A = \prod_{y \in V} (Y - A(y)) \tag{1}$$

and m_A is the squarefree part of χ_A. Finally, the trace $\mathrm{tr}(A)$ is, by definition, the trace of the endomorphism M_A; note that the trace is an $\mathbb{F}$-linear form.

4.1 Primitive representations

Primitive representations will allow us to work modulo I using only univariate polynomials. To start with, we say that $A \in R$ is a *primitive element* if the powers of A generate R. This is the case if and only if $\chi_A = m_A$; when I is radical, this is the case if and only if χ_A has no multiple root. In all that follows, we will be concerned only with primitive elements of the form $A = \sum_{i \leq s} \ell_i Y_i$ (as in many previous works, such as [20, 3, 21, 38, 22]). The following well-known result gives a condition on such an A to be a primitive element.

Lemma 8. *If I is radical, there exists a non-zero homogeneous polynomial Δ in $\mathbb{F}[L_1, \ldots, L_s]$ of degree less than $\delta^2/2$ such that if $\Delta(\ell_1, \ldots, \ell_s) \neq 0$, $A = \ell_1 Y_1 + \cdots + \ell_s Y_s$ is a primitive element.*

Proof. The argument is well-known: $A = \ell_1 Y_1 + \cdots + \ell_s Y_s$ is a primitive element if and only if the form $(y_1, \ldots, y_s) \mapsto \ell_1 y_1 + \cdots + \ell_1 y_s$ separates the zeros of I, that is, if $\ell_1(y_1 - y'_1) + \cdots + \ell_s(y_s - y'_s)$ is non-zero for all y and y' distinct zeros of I. Thus, it suffices to take for Δ the product of the linear forms $L_1(y_1 - y'_1) + \cdots + L_s(y_s - y'_s)$, for all pairs (y, y'); Δ has

coefficient in $\mathbb{F}$, as it is the square root of the discriminant of $\prod_{y\in V}(T - L_1y_1 - \cdots - L_sy_s)$, which has coefficients in $\mathbb{F}$. There are at most $\delta(\delta-1)/2$ such pairs (y, y'), and the conclusion follows. □

When $A = \sum_{i\le s} \ell_i Y_i$ is a primitive element, R and $\mathbb{F}[Y]/\langle P\rangle$ are isomorphic, with $P = m_A$; then, $\deg(P) = \delta$. In this case, a *primitive representation* $\mathscr{P} = (P, \mathbf{V}, \boldsymbol{\ell})$ contains the information necessary to encode this isomorphism: it consists of polynomials P and $\mathbf{V} = (V_1, \ldots, V_s)$ in $\mathbb{F}[Y]$, and $\boldsymbol{\ell} = (\ell_1, \ldots, \ell_s)$ in $\mathbb{F}^s$, with $\deg(V_i) < \delta$ for all i, such that the mappings

$$\begin{array}{rccc} \psi_{\mathscr{P}}: & R & \to & \mathbb{F}[Y]/\langle P\rangle \\ & Y_1, \ldots, Y_s & \mapsto & V_1, \ldots, V_s \end{array} \quad \text{and} \quad \begin{array}{rccc} \varphi_{\mathscr{P}}: & \mathbb{F}[Y]/\langle P\rangle & \to & R \\ & Y & \mapsto & \sum_{i\le s} \ell_i Y_i \end{array}$$

are isomorphisms, inverses of one another. In particular, $Y = \sum_{i\le s} \ell_i V_i$.

4.2 Mixed representations

We continue our discussion, with the purpose of introducing an intermediate data structure, between triangular sets and primitive representations.

We start with a variation on the notion of primitive element. For $j \le s$, let I_j be the ideal $I \cap \mathbb{F}[Y_1, \ldots, Y_j]$ and let R_j be the residue class ring $\mathbb{F}[Y_1, \ldots, Y_j]/I_j$.

We say that $A \in R$ is *a primitive element of level j* if A is in R_j, and if the powers of A generate R_j. The following lemma will be helpful to quantify linear forms that are primitive elements of level j; the proof is the same as that of Lemma 8.

Lemma 9. *Suppose that I is radical. Then for $j \le s$, there exists a non-zero homogeneous polynomial Δ_j in $\mathbb{F}[L_1, \ldots, L_j]$ of degree less than $\delta^2/2$ such that if $\Delta_j(\ell_1, \ldots, \ell_j) \ne 0$, $A = \ell_1Y_1 + \cdots + \ell_jY_j$ is a primitive element of level j.*

A *mixed representation* $\mathscr{M} = (\mathbf{P}, \mathbf{V}, \boldsymbol{\ell})$ of I of *format* $(j, s-j+1)$ consists in a triangular set $\mathbf{P} = (P, P_{j+1}, \ldots, P_s)$ in $\mathbb{F}[Y, Y_{j+1}, \ldots, Y_s]$, for the order $Y < Y_{j+1} < \cdots < Y_s$, some polynomials $\mathbf{V} = (V_1, \ldots, V_j)$ in $\mathbb{F}[Y]$ and $\boldsymbol{\ell} = (\ell_1, \ldots, \ell_j)$ in $\mathbb{F}^j$, such that we have mutually inverse isomorphisms

$$\begin{array}{rccc} \Psi_{\mathscr{M}}: & R & \to & R_{\mathbf{P}} \\ & Y_1, \ldots, Y_j & \mapsto & V_1, \ldots, V_j \\ & Y_{j+1}, \ldots, Y_s & \mapsto & Y_{j+1}, \ldots, Y_s. \end{array} \quad \text{and} \quad \begin{array}{rccc} \Phi_{\mathscr{M}}: & R_{\mathbf{P}} & \to & R \\ & Y & \mapsto & \sum_{i\le j} \ell_i Y_i \\ & Y_{j+1}, \ldots, Y_s & \mapsto & Y_{j+1}, \ldots, Y_s. \end{array}$$

In particular, $\sum_{i\le j} \ell_i V_i = Y$. Also, in this case, $\sum_{i\le j} \ell_i Y_i$ is a primitive element of level j, R_j is isomorphic to $\mathbb{F}[Y]/\langle P\rangle$, and $R_{j'}$ is isomorphic to $\mathbb{F}[Y, Y_{j+1}, \ldots, Y_{j'}]/\langle P, P_{j+1}, \ldots, P_{j'}\rangle$ for $j' > j$.

In other words, a mixed representation provides us with a primitive representation for the first j variables, and has a triangular shape for the last variables. The "format" $(j, s-j+1)$ provides a quick way to know how many elements are in $\mathbf{V}$ and $\boldsymbol{\ell}$ (here, j), and in $\mathbf{P}$ (here, $s - j + 1$). When $j = s$, a mixed representation is thus the same thing as a primitive representation. When $j = 1$, if we additionally suppose that $\ell_1 = 1$, we have $V_1 = Y$, so

up to renaming Y as Y_1, $\Psi_{\mathscr{M}}$ maps $Y_1, \ldots, Y_s$ to themselves, and $\mathbf{P}$ is a triangular set that generates I.

The following technical lemma will be useful in Section 6.

Lemma 10. *Suppose that I is radical and that $|\mathbb{F}| \geq \delta^2$. Then for $j \leq s$, the following are equivalent:*

- *the ideal I admits a mixed representation of format $(j, s-j+1)$*
- *for $i = j, \ldots, s-1$, there exists a positive integer d_{i+1} such that R_{i+1} is a free R_i-module with basis $1, Y_{i+1}, \ldots, Y_{i+1}^{d_{i+1}-1}$.*

Proof. Suppose that I admits a mixed representation $\mathscr{M} = (\mathbf{P}, \mathbf{V}, \boldsymbol{\ell})$ of format $(j, s-j+1)$, with polynomials $\mathbf{P} = (P, P_{j+1}, \ldots, P_s)$ in variables $Y, Y_{j+1}, \ldots, Y_s$. Through $\Psi_{\mathscr{M}}$, we see that for $i = j, \ldots, s-1$, R_{i+1} has the form $R_i[Y_{i+1}]/\langle P_{i+1}\rangle$; the second assertion in the lemma follows.

Conversely, we always have that $R_{i+1} = R_i[Y_{i+1}]/I_{i+1}$. Suppose R_{i+1} is a free R_i-module with basis $1, Y_{i+1}, \ldots, Y_{i+1}^{d_{i+1}-1}$. Then, there exists a polynomial P_{i+1} in $R_i[Y_{i+1}]$, monic of degree d_{i+1} in Y_{i+1}, that belongs to I_{i+1} (this is the characteristic polynomial of the multiplication by Y_{i+1}); one verifies that P_{i+1} actually generates I_{i+1} in $R_i[Y_{i+1}]$. As a consequence, we get that $R = R_j[Y_{j+1}, \ldots, Y_s]/\langle P_{j+1}, \ldots, P_s\rangle$. Using our assumption on the cardinality of $\mathbb{F}$, Lemma 9 ensures the existence of a primitive element of level j of the form $\ell_1 Y_1 + \cdots + \ell_j Y_j$ (since $\mathbb{F}$ is large enough, there must be a point in $\mathbb{F}^j$ where Δ_j does not vanish); this allows us to write $R_j \simeq \mathbb{F}[Y]/\langle P\rangle$, and the conclusion follows. □

4.3 Trace formulas

Finally, we describe how using trace formulas enables one to perform various conversions. The following claims are classical: see e.g. [43] for similar results using another linear form and [38] for a more general case, where I is not supposed to be radical.

Lemma 11. *Suppose that I is radical and let A and B be in R. Then, one can do the following in $\delta \operatorname{plog}(\delta)$ operations in $\mathbb{F}$:*

- *given $(\operatorname{tr}(A^j))_{j<2\delta}$, decide whether A is a primitive element, and if so, compute its minimal polynomial m_A;*
- *given m_A and $(\operatorname{tr}(BA^j))_{j<\delta}$, compute a polynomial $V \in \mathbb{F}[Y]$ of degree less than δ, such that if B can be expressed as a polynomial in A, then $B = V(A)$ in R.*

Remark that in the second item, we do not suppose that A is a primitive element, so that B may not be expressible in the form $V(A)$; the point is that if it is the case, we can find V. We do not include the cost of the verification that $B = V(A)$, since this would involve modular composition, which we do not know how to do in time $\delta \operatorname{plog}(\delta)$.

Proof. We start from the following classical formula (which is essentially a generating series version of Newton-Girard's identities)

$$\sum_{j\geq 0} \mathrm{tr}(A^{j+1})Y^j \in \mathbb{F}[[Y]] = -\frac{1}{\mathrm{rev}_\delta(\chi_A)} \frac{d\,\mathrm{rev}_\delta(\chi_A)}{dY}, \tag{2}$$

where we write $\mathrm{rev}_d(P) = Y^d P(1/Y)$ for any $P \in \mathbb{F}[Y]$ and $d \geq 0$. To recover χ_A using this equality, algorithms using Newton's formula (such as Rouillier's [38]) require divisions by integers $2, \ldots, \delta$, which may not be possible in small characteristic. Instead, we will use the Berlekamp-Massey algorithm; it allows us to compute the minimal polynomial μ_A of the sequence $(\mathrm{tr}(A^{j+1}))$ from the values $(\mathrm{tr}(A^j))_{j<2\delta}$.

Since I is radical, A is a primitive element if and only if χ_A has no multiple root, or equivalently if $\mathrm{rev}_\delta(\chi_A)$ has no multiple root, or equivalently if the rational function in Equation (2) is reduced. This is the case if and only if μ_A has degree δ; when this is the case, we have $\mu_A = m_A = \chi_A$. This proves the first point, since Berlekamp-Massey's algorithm runs in time $\delta\,\mathrm{plog}(\delta)$.

The second point is in [43, Theorem 5], up to an inconsequential difference (that result is proved using another linear form than the trace). □

The previous lemma allows one to compute a primitive representation by means of trace computations. We now discuss how to compute a triangular representation. While the idea of using trace formulas remains, the computations are more involved. For this reason, we consider only bivariate situations. The following result is from [35, Section 3]; it requires a stronger assumption on the characteristic than the previous lemma (as we use a bivariate version of Newton's identities). This assumption may most likely be lifted, but we leave this generalization to future work.

Lemma 12. *Suppose that $I \subset \mathbb{F}[Y_1, Y_2]$ is radical and let p be the characteristic of $\mathbb{F}$. If $p > \delta$, then one can do the following using $\delta\,\mathrm{plog}(\delta)$ operations in $\mathbb{F}$:*

- *given $(\mathrm{tr}(Y_1^j))_{j<\delta}$, verify whether I is generated by a triangular set $(T_1(Y_1), T_2(Y_1, Y_2))$, and if so compute T_1;*
- *given T_1 and $(\mathrm{tr}(Y_1^i Y_2^j))_{i<d_1, j<d_2}$, with $d_1 = \deg(T_1)$ and $d_2 = \delta/d_1$, compute T_2.*

5 Proof of Theorem 1

We will now prove our first main theorem, on the cost of multiplication, inversion, norm computation, modular composition and power projection modulo a triangular set. The algorithms in this section will solve these problems by computing a primitive representation, since the questions mentioned in Theorem 1 can all be solved in quasi-linear time for univariate polynomials.

The basic idea to convert to a primitive representation uses trace formulas (by means of Lemma 11); it mainly amounts to solving some instances of power projection and modular composition. The delicate question is how to perform efficiently these power projections,

or modular compositions. Section 3 gave algorithms that are efficient when the number of variables s is fixed, but not when s is allowed to grow (recall that the estimates of that section hide an exponential dependency in s).

To solve this issue, we will not proceed directly. The key step is to first solve the problem for $s = 2$, that is, for bivariate triangular sets, as this alleviates the issue of the exponential cost in s; this is done in Subsection 5.2. For higher values of s, with $\mathbf{T} = (T_1, \ldots, T_s)$, we will then first deal with (T_1, T_2), finding a univariate polynomial P such that $\mathbb{F}_q[Y_1, Y_2]/\langle T_1, T_2\rangle \simeq \mathbb{F}_q[Y]/\langle P\rangle$, then continue with (P, T_3), and so on. This is done in Subsection 5.3. The proof of Theorem 1 is then given in Subsection 5.4. First, though, we show the details of our conversion algorithm on an example in three variables.

5.1 A worked example

The following example, with $s = 3$ over $\mathbb{F}_{101}$, will be used in this section and in the next one. We start from $\mathbf{T} = (T_1, T_2, T_3)$ given by

$$\mathbf{T} \left| \begin{array}{l} T_3 = Y_3^2 + 100Y_1 \\ T_2 = Y_2^2 + Y_1 \\ T_1 = Y_1^2 + 1. \end{array} \right.$$

We will show how to establish that $Y_1 + Y_2 + Y_3$ is a generator of $R_{\mathbf{T}} = \mathbb{F}_{101}[Y_1, Y_2, Y_3]/\langle \mathbf{T}\rangle$, as well as expressions for Y_1, Y_2, Y_3 in terms of Y.

As said before, we do not proceed directly: the key is to first solve the problem for (T_1, T_2). This is done by introducing the linear combination $Y_1 + Y_2$, and using a bivariate change of order algorithm (coming from Lemma 11) to yield the isomorphism

$$\begin{array}{ccc} \mathbb{F}_{101}[Y_1, Y_2]/\langle T_1, T_2\rangle & \to & \mathbb{F}_{101}[Y]/\langle P\rangle \\ Y_1 & \mapsto & V_1 \\ Y_2 & \mapsto & V_2 \\ Y_1 + Y_2 & \leftarrow\!\shortmid & Y, \end{array}$$

with $P = Y^4 + 2Y^2 + 97Y + 2$ and

$$\begin{aligned} V_1 &= 68Y^3 + 34Y^2 + 2Y + 32 \\ V_2 &= 33Y^3 + 67Y^2 + 100Y + 69. \end{aligned}$$

Re-introducing Y_3, this can be readily extended to the following isomorphism

$$\begin{array}{ccc} \mathbb{F}_{101}[Y_1, Y_2, Y_3]/\langle \mathbf{T}\rangle & \to & \mathbb{F}_{101}[Y, Y_3]/\langle \mathbf{T}'\rangle \\ Y_1 & \mapsto & V_1 \\ Y_2 & \mapsto & V_2 \\ Y_3 & \mapsto & Y_3 \\ Y_1 + Y_2 & \leftarrow\!\shortmid & Y, \end{array}$$

where $\mathbf{T}'$ is the bivariate triangular set $(P(Y), T_3(V_1, V_2, Y_3))$ in $\mathbb{F}_{101}[Y, Y_3]$. Next, we apply

the bivariate algorithm to $\mathbf{T}'$; this gives the isomorphism

$$\begin{array}{ccc} \mathbb{F}_{101}[Y,Y_3]/\langle \mathbf{T}'\rangle & \to & \mathbb{F}_{101}[Z]/\langle P'\rangle \\ Y & \mapsto & V' \\ Y_3 & \mapsto & V_3' \\ Y+Y_3 & \leftarrow\!\shortmid & Z, \end{array}$$

with this time $P' = Z^8 + 4Z^6 + 99Z^4 + 52Z^2 + 9$ and

$$\begin{aligned} V' &= 51Z^7 + 30Z^6 + 32Z^5 + 30Z^4 + 48Z^3 + 14Z^2 + 3Z + 78 \\ V_3' &= 50Z^7 + 71Z^6 + 69Z^5 + 71Z^4 + 53Z^3 + 87Z^2 + 99Z + 23. \end{aligned}$$

Composing the previous results, this leads to the following isomorphism, which is what we were looking for:

$$\begin{array}{ccc} \mathbb{F}_{101}[Y_1,Y_2,Y_3]/\langle \mathbf{T}\rangle & \to & \mathbb{F}_{101}[Z]/\langle P'\rangle \\ Y_1 & \mapsto & W_1 \\ Y_2 & \mapsto & W_2 \\ Y_3 & \mapsto & W_3 \\ Y_1+Y_2+Y_3 & \leftarrow\!\shortmid & Z, \end{array}$$

with

$$\begin{aligned} W_1 &= V_1(V') \bmod P' &&= Z^7 + 64Z^5 + 96Z^3 + 5Z \\ W_2 &= V_2(V') \bmod P' &&= 50Z^7 + 30Z^6 + 69Z^5 + 30Z^4 + 53Z^3 + 14Z^2 + 99Z + 78 \\ W_3 &= V_3' &&= 50Z^7 + 71Z^6 + 69Z^5 + 71Z^4 + 53Z^3 + 87Z^2 + 99Z + 23. \end{aligned}$$

5.2 The bivariate case

In this subsection, we handle the bivariate case only. Let $\mathbb{F}_q$ be our base field and consider a triangular set $\mathbf{T} = (T_1, T_2)$ in $\mathbb{F}_q[Y_1, Y_2]$, of multidegree $\mathbf{d} = (d_1, d_2)$. In the following proposition, we give cost estimates on the computation of a primitive representation $\mathscr{P} = (P, \boldsymbol{\ell}, \mathbf{V})$, and on the cost of applying $\psi_{\mathscr{P}} : R_{\mathbf{T}} \to \mathbb{F}_q[Y]/\langle P\rangle$ and its inverse $\varphi_{\mathscr{P}}$. We must make the assumption that q is large enough, so as to be sure that there exist enough primitive elements of the requested form. Then, the algorithm to find $\mathscr{P}$ is Las Vegas: we choose the candidate primitive element at random, but we can always verify whether our choice is correct.

We choose additionally to set one of the ℓ_i to 1, as this will be useful in the next section.

Proposition 13. *For $\varepsilon > 0$, one can do the following in an expected $\delta_{\mathbf{d}}^{1+\varepsilon}\log(q)\operatorname{plog}_\varepsilon(\log(q))$ bit operations: for $\mathbf{T} = (T_1, T_2)$ of multidegree $\mathbf{d} = (d_1, d_2)$ in $\mathbb{F}_q[Y_1, Y_2]$, such that $q \geq \delta_{\mathbf{d}}^2$,*

- *determine whether $\mathbf{T}$ is squarefree;*
- *if so, after choosing either $\ell_1 = 1$ or $\ell_2 = 1$, compute a primitive representation $\mathscr{P} = (P, \mathbf{V}, \boldsymbol{\ell})$ of $\mathbf{T}$, with $\boldsymbol{\ell} = (\ell_1, \ell_2)$.*

With notation as before, one can do the following in $\delta_{\mathbf{d}}^{1+\varepsilon}\log(q)\operatorname{plog}_\varepsilon(\log(q))$ bit operations:

- *given $\mathscr{P}$ and B in $R_{\mathbf{T}}$, compute $\psi_{\mathscr{P}}(B) \in \mathbb{F}_q[Y]/\langle P\rangle$;*
- *given $\mathscr{P}$ and B in $\mathbb{F}_q[Y]/\langle P\rangle$, compute $\varphi_{\mathscr{P}}(B) \in R_{\mathbf{T}}$.*

Proof. To test whether $\mathbf{T}$ generates a radical ideal, it is enough to compute the gcd of T_1 and dT_1/dY_1, as well as a gcd of T_2 and $\partial T_2/\partial Y_2$ modulo T_1, and check whether all are constant. The first computation is a simple application of the half-gcd algorithm, and takes time $d_1 \operatorname{plog}(d_1)$. The second one is more delicate, as it involves the half-gcd algorithm with coefficients modulo T_1, and T_1 may not be irreducible: this question is treated in [1], with an algorithm of cost $\delta_{\mathbf{d}} \operatorname{plog}(\delta_{\mathbf{d}})$, with $\delta_{\mathbf{d}} = d_1 d_2$. This settles the first point.

To determine whether $A = \ell_1 Y_1 + \ell_2 Y_2$ is a primitive element, we compute the traces $(\operatorname{tr}(A^j))_{j<2\delta_{\mathbf{d}}}$ and apply Lemma 11. Computing these traces requires to first compute the traces of the monomial basis modulo $\langle \mathbf{T}\rangle$: it is shown in [35] that this can be done in quasi-linear time. Then, we are left with an instance of power projection with parameter $\mathbf{e} = (2\delta_{\mathbf{d}})$. Invoking Theorem 3, this takes $\delta_{\mathbf{d}}^{1+\varepsilon} \log(q) \operatorname{plog}_\varepsilon(\log(q))$ bit operations; the other $\delta_{\mathbf{d}} \operatorname{plog}(\delta_{\mathbf{d}})$ $\mathbb{F}_q$-operations that appear in Lemma 11 are not more expensive. Because $q \geq \delta_{\mathbf{d}}^2$, Lemma 8 shows that at least half of the linear combinations $A = \ell_1 Y_1 + \ell_2 Y_2$, with either $\ell_1 = 1$ or $\ell_2 = 1$, are primitive elements. Thus, we expect to have to go through this process at most twice.

If A is a primitive element, we continue by computing $\operatorname{tr}(Y_1 A^i)_{i<\delta_{\mathbf{d}}}$ and $\operatorname{tr}(Y_2 A^i)_{i<\delta_{\mathbf{d}}}$; again, this is done by invoking Theorem 3. From these values, Lemma 11 shows how to deduce V_1 and V_2 in quasi-linear time. Thus, we have obtained $\mathscr{P} = (P, (V_1, V_2), \boldsymbol{\ell})$, proving the second point.

Given $\mathscr{P}$ and B in $R_{\mathbf{T}}$, computing $\psi_{\mathscr{P}}(B)$ amounts to computing $B(V_1, V_2) \bmod P$. This is an instance of modular composition modulo P with parameter $\mathbf{d} \in \mathbb{N}^2$, so it can be done in $\delta_{\mathbf{d}}^{1+\varepsilon} \log(q) \operatorname{plog}_\varepsilon(\log(q))$ bit operations by Theorem 3. This proves the third point.

Finally, given $\mathscr{P}$ and B in $\mathbb{F}_q[Y]/\langle P\rangle$, computing $\varphi_{\mathscr{P}}(B)$ amounts to computing $B(\ell_1 Y_1 + \ell_2 Y_2) \bmod \langle \mathbf{T}\rangle$. This is an instance of modular composition modulo $\langle \mathbf{T}\rangle$ with parameter $(\delta_{\mathbf{d}}) \in \mathbb{N}^1$; it can be done in $\delta_{\mathbf{d}}^{1+\varepsilon} \log(q) \operatorname{plog}_\varepsilon(\log(q))$ bit operations by Theorem 3. This proves the last point. □

5.3 The general case

We will now extend the former construction to a higher number of variables. Suppose thus that s is arbitrary, and let $\mathbf{T} = (T_1, \ldots, T_s)$ be a triangular set of multidegree $\mathbf{d} = (d_1, \ldots, d_s)$ in $\mathbb{F}_q[\mathbf{Y}]$, with $\mathbf{Y} = Y_1, \ldots, Y_s$. As explained before, our idea is to deal first with (T_1, T_2), and continue this way until there is only one variable left. We start from a primitive representation $\mathscr{P} = (P, (V_1, V_2), (\ell_1, \ell_2))$ of (T_1, T_2); as in the previous subsection, we choose to add the constraint that either $\ell_1 = 1$ or $\ell_2 = 1$, for future use. We can then define

$$\mathbf{P} = (P,\ P_3 = T_3(V_1, V_2, Y_3) \bmod P,\ \ldots,\ P_s = T_s(V_1, V_2, Y_3, \ldots, Y_s) \bmod P).$$

This is a triangular set in $\mathbb{F}_q[Y, Y_3, \ldots, Y_s]$ of multidegree $\mathbf{d}' = (d_1 d_2, d_3, \ldots, d_s) \in \mathbb{N}^{s-1}$. It follows from this construction that we have the following isomorphisms, inverse of one

another:

$$\begin{array}{ccc} R_{\mathbf{T}} & \to & R_{\mathbf{P}} \\ Y_1, Y_2 & \mapsto & V_1, V_2 \\ Y_3, \ldots, Y_s & \mapsto & Y_3, \ldots, Y_s \end{array} \quad \text{and} \quad \begin{array}{ccc} R_{\mathbf{P}} & \to & R_{\mathbf{T}} \\ Y & \mapsto & \ell_1 Y_1 + \ell_2 Y_2 \\ Y_3, \ldots, Y_s & \mapsto & Y_3, \ldots, Y_s. \end{array}$$

In other words, we have obtained a mixed representation $\mathscr{M}$ of format $(2, s-1)$ of $\mathbf{T}$, and the mappings above are none other than the isomorphisms $\Psi_{\mathscr{M}}$ and $\Phi_{\mathscr{M}}$ associated with it. The following lemma summarizes all the costs involved.

Lemma 14. *Fix $\varepsilon > 0$. Then, for $\mathbf{d}$ in $\mathbb{N}^s$ and $\mathbf{T} = (T_1, \ldots, T_s)$ of multidegree $\mathbf{d} = (d_1, \ldots, d_s)$ in $\mathbb{F}_q[Y_1, \ldots, Y_s]$, such that $q \geq \delta_{\mathbf{d}}^2$, one can do the following in an expected $s\,\delta_{\mathbf{d}}^{1+\varepsilon} \log(q)\, \mathrm{plog}_\varepsilon(\log(q))$ bit operations:*

- *determine whether $\langle T_1, T_2 \rangle$ is a radical ideal;*
- *if so, after choosing either $\ell_1 = 1$ or $\ell_2 = 1$, compute a mixed representation $\mathscr{M} = (\mathbf{P}, \mathbf{V}, \boldsymbol{\ell})$ of $\mathbf{T}$ of format $(2, s-1)$, with $\boldsymbol{\ell} = (\ell_1, \ell_2)$.*

With notation as before, one can do the following in $\delta_{\mathbf{d}}^{1+\varepsilon} \log(q)\, \mathrm{plog}_\varepsilon(\log(q))$ bit operations:

- *given $\mathscr{M}$ and B in $R_{\mathbf{T}}$, compute $\Psi_{\mathscr{M}}(B) \in R_{\mathbf{P}}$;*
- *given $\mathscr{M}$ and B in $R_{\mathbf{P}}$, compute $\Phi_{\mathscr{M}}(B) \in R_{\mathbf{T}}$.*

Proof. The first point was proved in Proposition 13, as well as the estimate on the cost of computing a primitive representation $\mathscr{P} = (P, (V_1, V_2), (\ell_1, \ell_2))$ of (T_1, T_2), with either $\ell_1 = 1$ or $\ell_2 = 1$. To compute all other polynomials in $\mathbf{P}$, we need to apply some modular compositions: for $i \leq s$, P_i is obtained applying $\psi_{\mathscr{P}}$ to all coefficients of T_i, assuming we view T_i as a polynomial in $Y_3, \ldots, Y_i$. This requires $d_3 \cdots d_i$ applications of $\psi_{\mathscr{P}}$ for T_i, for a total of $d_3 + \cdots + d_3 \cdots d_s \leq s d_3 \cdots d_s$ applications. Each application takes time $(d_1 d_2)^{1+\varepsilon} \log(q)\, \mathrm{plog}_\varepsilon(\log(q))$ bit operations by Proposition 13. This proves the second point, since $(d_1 d_2)(d_3 \cdots d_s) = \delta_{\mathbf{d}}$. The third and fourth points are proved similarly. □

Our idea is now straightforward: continue is a similar manner with $\mathbf{P}$, introducing a primitive representation for (P, P_3), until we are left with univariate polynomials.

Proposition 15. *Fix $\varepsilon > 0$. Then, for $\mathbf{d}$ in $\mathbb{N}^s$ and $\mathbf{T} = (T_1, \ldots, T_s)$ of multidegree $\mathbf{d} = (d_1, \ldots, d_s)$ in $\mathbb{F}_q[Y_1, \ldots, Y_s]$, such that $q \geq \delta_{\mathbf{d}}^2$, one can do the following in an expected $s^2\,\delta_{\mathbf{d}}^{1+\varepsilon} \log(q)\, \mathrm{plog}_\varepsilon(\log(q))$ bit operations:*

- *determine whether $\mathbf{T}$ is a radical ideal;*
- *if so, compute $\mathscr{M}_{s-1}, \ldots, \mathscr{M}_1$, with $\mathscr{M}_i = (\mathbf{P}_i, \mathbf{V}_i, \boldsymbol{\ell}_i)$, such that*
  - *$\mathscr{M}_{s-1}$ is a mixed representation of $\mathbf{T}$, of format $(2, s-1)$,*
  - *for $i = s-2, \ldots, 1$, $\mathscr{M}_i$ is a mixed representation of $\mathbf{P}_{i+1}$, of format $(2, i)$,*
  - *for $i = s-1, \ldots, 1$, $\boldsymbol{\ell}_i = (\ell_{i,1}, \ell_{i,2})$, with either $\ell_{i,1} = 1$ or $\ell_{i,2} = 1$.*

With notation as before, let P be the minimal polynomial of $\mathscr{M}_1$, let $\Psi = \Psi_{\mathscr{M}_1} \circ \cdots \circ \Psi_{\mathscr{M}_{s-1}}$ and let Φ be its inverse. Then, one can do the following in $s\,\delta_{\mathbf{d}}^{1+\varepsilon}\log(q)\,\mathrm{plog}_\varepsilon(\log(q))$ bit operations:

- *given $\mathscr{M}_{s-1}, \ldots, \mathscr{M}_1$ and B in $R_{\mathbf{T}}$, compute $\Psi(B) \in \mathbb{F}_q[Y]/\langle P\rangle$;*
- *given $\mathscr{M}_{s-1}, \ldots, \mathscr{M}_1$ and B in $\mathbb{F}_q[Y]/\langle P\rangle$, compute $\Phi(B) \in R_{\mathbf{T}}$.*

Finally, one can do the following in $s^2\,\delta_{\mathbf{d}}^{1+\varepsilon}\log(q)\,\mathrm{plog}_\varepsilon(\log(q))$ bit operations:

- *given $\mathscr{M}_{s-1}, \ldots, \mathscr{M}_1$, compute a primitive representation $\mathscr{P} = (P, \mathbf{V}, \boldsymbol{\ell})$ for $\mathbf{T}$ such that $\Psi = \Psi_{\mathscr{P}}$ and $\Phi = \Phi_{\mathscr{P}}$.*

Besides, if we choose a priori $j \le s$, one can ensure that $\boldsymbol{\ell} = (\ell_1, \ldots, \ell_s)$ is such that $\ell_j = 1$.

Proof. To compute $\mathscr{M}_{s-1}, \ldots, \mathscr{M}_1$, it suffices to apply s times Lemma 14. The cost of applying Ψ and Φ follows similarly from Lemma 14. At this point, to determine $\mathscr{P} = (P, \mathbf{V}, \boldsymbol{\ell})$, it suffices to compute $\mathbf{V} = (V_1, \ldots, V_s)$ and $\boldsymbol{\ell} = (\ell_1, \ldots, \ell_s)$. The V_i are obtained by computing $\Psi(Y_i)$, thus requiring s applications of Ψ.

Finally, to compute $(\ell_1, \ldots, \ell_s)$, and ensure $\ell_j = 1$, let us determine the image of Y by Φ: a quick check shows that it is given by $Y \mapsto \ell_{1,1}\cdots\ell_{s-1,1}Y_1 + \ell_{1,1}\cdots\ell_{s-2,1}\ell_{s-1,2}Y_2 + \cdots + \ell_{1,1}\ell_{2,1}\ell_{3,2}Y_{s-2} + \ell_{1,1}\ell_{2,2}Y_{s-1} + \ell_{1,2}Y_s$, so that the coefficient of Y_j is $\ell_{1,1}\cdots\ell_{s-j,1}\ell_{s-j+1,2}$ (where undefined terms are set to 1). Choosing $\ell_{1,1} = \cdots = \ell_{s-j,1} = 1$ and $\ell_{s-j+1,2} = \cdots = \ell_{s-1,2} = 1$ ensures that this coefficient equals 1. □

5.4 Proof of Theorem 1

We will finally use the former results to solve the questions stated in Theorem 1; in all that follows, $\varepsilon > 0$ is fixed. We start from a triangular set $\mathbf{T}$ of multidegree $\mathbf{d} = (d_1, \ldots, d_s)$ in $\mathbb{F}_q[Y_1, \ldots, Y_s]$. To solve the questions of Theorem 1, we will want to apply Proposition 15. To apply this result, we need to ensure that the base field $\mathbb{F}_q$ has cardinality at least $\delta_{\mathbf{d}}^2$:

- if $q \ge \delta_{\mathbf{d}}^2$, we just let $q' = q$,
- if $q < \delta_{\mathbf{d}}^2$, we use fact F_3 to build an extension $\mathbb{F}_{q'}$ of $\mathbb{F}_q$ of degree $\lceil \log_q(\delta_{\mathbf{d}}^2)\rceil$, and embed all coefficients of $\mathbf{T}$ in $\mathbb{F}_{q'}$; remark that $q' \le q\delta_{\mathbf{d}}^2 \le \delta_{\mathbf{d}}^4$. From fact F_3, the cost of finding the embedding is $\sqrt{p}\,\mathrm{plog}(q')$ $\mathbb{F}_p$-operations, where p is the characteristic of $\mathbb{F}_q$; this fits into the bound $\delta_{\mathbf{d}}\,\mathrm{plog}(\delta_{\mathbf{d}})$ $\mathbb{F}_p$-operations. The cost of mapping an element of $\mathbb{F}_q$ to $\mathbb{F}_{q'}$, and back, is $\mathrm{plog}(\delta_{\mathbf{d}})$ $\mathbb{F}_p$-operations. Thus, in all problems, the costs of all embeddings and of all back conversions will fit into the bound $\delta_{\mathbf{d}}\,\mathrm{plog}(\delta_{\mathbf{d}})$, or $(\delta_{\mathbf{d}} + \delta_{\mathbf{e}})\,\mathrm{plog}(\delta_{\mathbf{d}})$ for modular composition and power projection.

From now on, our base field is $\mathbb{F}_{q'}$. For all questions below, we start by testing whether $\mathbf{T}$ is squarefree, and if so, we compute a primitive representation $\mathscr{P} = (P, \mathbf{V}, \boldsymbol{\ell})$ of $\mathbf{T}$ (over $\mathbb{F}_{q'}$). By Proposition 15, this can be done in an expected $s^2\,\delta_{\mathbf{d}}^{1+\varepsilon}\log(q')\,\mathrm{plog}_\varepsilon(\log(q'))$ bit operations. Applying $\Psi_{\mathscr{P}}$ and its inverse can then be done in time $s\,\delta_{\mathbf{d}}^{1+\varepsilon}\log(q')\,\mathrm{plog}_\varepsilon(\log(q'))$, by the same proposition. This setup allows us to perform operations in $R_{\mathbf{T}}$ by mapping to $\mathbb{F}_{q'}[Y]/\langle P\rangle$, solving the univariate problem, and mapping back to $R_{\mathbf{T}}$:

- Multiplication is straightforward, since multiplication modulo P can be done in $\delta_{\mathbf{d}} \operatorname{plog}(\delta_{\mathbf{d}})$ operations in $\mathbb{F}_{q'}$.

- The same holds for inversion: to test whether $A \in R_{\mathbf{T}}$ is a unit, and invert it if possible, we will attempt to invert $A' = \Psi_{\mathscr{P}}(A)$ modulo P, and pull back the inverse. This amounts to computing the extended gcd of A' and P: A is a unit in $R_{\mathbf{T}}$ if and only if this gcd is 1, in which case the cofactor provides the desired inverse. The cost of extended gcd computation is again $\delta_{\mathbf{d}} \operatorname{plog}(\delta_{\mathbf{d}})$ operations in $\mathbb{F}_{q'}$.

- Computing the norm of A modulo $\langle \mathbf{T} \rangle$ works in a similar manner. Recall that the norm is the determinant of the endomorphism M_A of multiplication by A modulo $\langle \mathbf{T} \rangle$. Given $A' = \Psi_{\mathscr{P}}(A)$, the norm of A is thus given by the resultant of A' and P. The cost of computing this resultant is $\delta_{\mathbf{d}} \operatorname{plog}(\delta_{\mathbf{d}})$ operations in $\mathbb{F}_{q'}$.

- We consider next modular composition: given $(G_1, \ldots, G_m)$ in $R_{\mathbf{T}}^m$, $\mathbf{e} \in \mathbb{N}^m$ and F in $\mathbb{F}_q[X_1, \ldots, X_m]_{\mathbf{e}}$, we want to compute $F(G_1, \ldots, G_m) \in R_{\mathbf{T}}$. As in the theorem, we will work under the assumption that $m \leq 2$. Then, we get our result by computing

$$\Phi_{\mathscr{P}}\big(F(G'_1, \ldots, G'_m) \bmod P\big),$$

  with $G'_i = \Psi_{\mathscr{P}}(G_i)$. This requires $m+1 \leq 3$ applications of $\Phi_{\mathscr{P}}$ or $\Psi_{\mathscr{P}}$, and a modular composition modulo P. The latter can be done in $(\delta_{\mathbf{d}} + \delta_{\mathbf{e}})^{1+\varepsilon} \log(q') \operatorname{plog}_{\varepsilon}(\log(q'))$ bit operations by Theorem 3.

- We use a similar strategy for power projection. Given $(G_1, \ldots, G_m)$ in $R_{\mathbf{T}}^m$, $\mathbf{e} \in \mathbb{N}^m$ and τ in $R_{\mathbf{T}}^*$, we want to compute the values $\tau(G_1^{a_1} \cdots G_m^{a_m})$, for $0 \leq a_i < e_i$, $i \leq m$. Again, we suppose that $m \leq 2$. Then, the algorithm is the transpose of the one for modular composition: we compute all G'_i as above, then the linear form $\tau' = \Phi^t_{\mathscr{P}}(\tau)$ defined modulo P, and obtain our result by computing all $\tau'({G'_1}^{a_1} \cdots {G'_m}^{a_m})$ by univariate power projection.

  The cost analysis is the same as before; the only missing ingredient is the cost of applying the transpose map $\Phi^t_{\mathscr{P}}$. It is however straightforward to transpose the algorithm we gave for Φ. Indeed, this algorithm boils down to $s-1$ applications of maps of the form $\Phi_{\mathscr{M}_i}$. Each of them is done through modular composition with $m = 1$ and $s = 2$. The transposed map uses power projection with the same parameters; using Theorem 3, we obtain the same asymptotic estimate as in the forward direction.

Summing all contributions, and using the upper bound $\operatorname{plog}(\delta_{\mathbf{d}}) \leq c\delta_{\mathbf{d}}^{\varepsilon}$, for some constant c depending on ε, we see that all costs are of the form $s^2\, \delta_{\mathbf{d}}^{1+\varepsilon} \log(q') \operatorname{plog}_{\varepsilon}(\log(q'))$, or $s^2\, (\delta_{\mathbf{d}} + \delta_{\mathbf{e}})^{1+\varepsilon} \log(q') \operatorname{plog}_{\varepsilon}(\log(q'))$ for those involving a parameter $\mathbf{e}$.

It remains to express these estimates in terms of $\log(q)$ instead of $\log(q')$. In any case, $\log(q')$ is at most $\log(q) + 2\log(\delta_{\mathbf{d}})$, so that any cost of the form $\log(q') \operatorname{plog}_{\varepsilon}(\log(q'))$ is actually in $\delta_{\mathbf{d}}^{\varepsilon} \log(q) \operatorname{plog}_{\varepsilon}(\log(q))$. Up to replacing ε by $\varepsilon/2$ everywhere, this proves Theorem 1.

6 Proof of Theorem 2

We will now answer the last question, change of order, thereby proving Theorem 2. In the previous section, the approach consisted in introducing successive mixed representations to progressively convert from a triangular representation to a univariate one; here, we will go in the opposite direction. As before, we start with a worked example. Then, we deal with the algorithm in the bivariate case, and extend the results to an arbitrary number of variables in a second step.

6.1 A worked example

We first illustrate our strategy on the example of the previous section. We are given the triangular set

$$\mathbf{T} \left| \begin{array}{l} Y_3^2 + 100Y_1 \\ Y_2^2 + Y_1 \\ Y_1^2 + 1, \end{array} \right.$$

for the order $Y_1 < Y_2 < Y_3$, defined over $\mathbb{F}_{101}$. We will show how to determine a triangular set $\mathbf{T}'$ for the order $Y_3 < Y_1 < Y_2$ that generates the same ideal as $\mathbf{T}$. In the previous section, up to a slight change of notation, we obtained the isomorphism

$$\begin{array}{ccc} \mathbb{F}_{101}[Y_1, Y_2, Y_3]/\langle \mathbf{T} \rangle & \to & \mathbb{F}_{101}[Y]/\langle P \rangle \\ Y_1 & \mapsto & V_1 \\ Y_2 & \mapsto & V_2 \\ Y_3 & \mapsto & V_3 \end{array}$$

with $P = Y^8 + 4Y^6 + 99Y^4 + 52Y^2 + 9$ and

$$\begin{aligned} V_1 &= Y^7 + 64Y^5 + 96Y^3 + 5Y \\ V_2 &= 50Y^7 + 30Y^6 + 69Y^5 + 30Y^4 + 53Y^3 + 14Y^2 + 99Y + 78 \\ V_3 &= 50Y^7 + 71Y^6 + 69Y^5 + 71Y^4 + 53Y^3 + 87Y^2 + 99Y + 23. \end{aligned}$$

This will be our starting point here; all operations that follow are either bivariate change of orders, or modular compositions.

Introducing Y_2. Let $A = V_1 + V_3$: this plays the role of a "random" linear combination of V_1, V_3, that correspond to the lowest variables for the new order. Since A has been chosen "random" enough, we can express V_1 and V_3 as polynomials in A: Lemma 11 gives us two polynomials V_1' and V_3' in $\mathbb{F}_{101}[Z]$ such that $V_1 = V_1'(A) \bmod P$ and $V_3 = V_3'(A) \bmod P$. Explicitly, we have $V_1' = 68Z^3 + 67Z^2 + 2Z + 69$ and $V_3' = 33Z^3 + 34Z^2 + 100Z + 32$.

Consider now the polynomials $(P(Y), Z - A(Y))$, which form a triangular set for the order $Y < Z$. Using trace computations as in Lemma 12, we can determine a triangular set $(Q(Z), R(Z, Y))$ for the order $Z < Y$, that generates the same ideal; explicitly, we get

$$\left| \begin{array}{l} R = Y^2 + 99YZ + 68Z^3 + 68Z^2 + 2Z + 69 \\ Q = Z^4 + 2Z^2 + 4Z + 2. \end{array} \right.$$

At this point, we have thus determined an isomorphism

$$\begin{array}{ccc} \mathbb{F}_{101}[Y]/\langle P\rangle & \to & \mathbb{F}_{101}[Z,Y]/\langle Q(Z), R(Z,Y)\rangle \\ Y & \mapsto & Y \\ A & \mapsto & Z \end{array}$$

which maps V_1 to V_1' and V_3 to V_3'. Remembering that $Y = V_1 + V_2 + V_3 = A + V_2$, we deduce that the image of V_2 is $Y - Z$.

This leads us to define $S = R(Z, Y + Z) \bmod Q$, or, explicitly, $S = Y^2 + 68Z^3 + 67Z^2 + 2Z + 69$. Then, we get the isomorphism

$$\begin{array}{ccc} \mathbb{F}_{101}[Y]/\langle P\rangle & \to & \mathbb{F}_{101}[Z,Y]/\langle Q(Z), S(Z,Y)\rangle \\ Y & \mapsto & Y + Z \\ A & \mapsto & Z \\ V_2 & \mapsto & Y. \end{array}$$

Remembering that $\mathbb{F}_{101}[Y]/\langle P\rangle$ is isomorphic to $\mathbb{F}_{101}[Y_1, Y_2, Y_3]/\langle \mathbf{T}\rangle$, we finally obtain

$$\begin{array}{ccc} \mathbb{F}_{101}[Y_1, Y_2, Y_3]/\langle \mathbf{T}\rangle & \to & \mathbb{F}_{101}[Z,Y]/\langle Q(Z), S(Z,Y)\rangle \\ Y_1 & \mapsto & V_1'(Z) \\ Y_2 & \mapsto & Y \\ Y_3 & \mapsto & V_3'(Z). \end{array}$$

At this stage, we have obtained a representation by means of the bivariate triangular set $(Q(Z), S(Z, Y))$, with $Y \simeq Y_2$ as highest variable.

Introducing Y_1. To reintroduce Y_1, the process is similar, except that we are left to work modulo Q. Consider the triangular set $(Q(Z), T - V_3'(Z))$, and perform as before a bivariate change of order; we obtain $(F(T), G(T, Z))$, with

$$\left| \begin{array}{l} G = Z + 100T^2 + 100T \\ F = T^4 + 1. \end{array} \right.$$

Thus, we have the isomorphism

$$\begin{array}{ccc} \mathbb{F}_{101}[Z]/Q & \to & \mathbb{F}_{101}[T,Z]/\langle F(T), G(T,Z)\rangle \\ Z & \mapsto & Z \\ V_3' & \mapsto & T; \end{array}$$

since $Z = V_1' + V_3'$, the image of V_1' is $Z - T$. As before, this leads us to introduce $H = G(T, Z + T) \bmod F = Z + 100T^2$. This gives us an isomorphism

$$\begin{array}{ccc} \mathbb{F}_{101}[Z]/Q & \to & \mathbb{F}_{101}[T,Z]/\langle F(T), H(T,Z)\rangle \\ Z & \mapsto & Z + T \\ V_3' & \mapsto & T \\ V_1' & \mapsto & Z. \end{array}$$

Finally, we let $K(T, Z, Y)$ be obtained by applying the former map coefficient-wise to $S(Z, Y)$; explicitly, we obtain the polynomial $K(T, Z, Y) = Y^2 + T^2$, and the isomorphism

$$\begin{array}{ccc} \mathbb{F}_{101}[Z,Y]/\langle Q(Z), S(Z,Y)\rangle & \to & \mathbb{F}_{101}[T,Z,Y]/\langle F(T), H(T,Z), K(T,Z,Y)\rangle \\ V_3' & \mapsto & T \\ V_1' & \mapsto & Z \\ Y & \mapsto & Y. \end{array}$$

Taking into account the result of the previous paragraph, we deduce by composition the isomorphism

$$\begin{array}{ccc} \mathbb{F}_{101}[Y_1,Y_2,Y_3]/\langle \mathbf{T}\rangle & \to & \mathbb{F}_{101}[T,Z,Y]/\langle F(T), H(T,Z), K(T,Z,Y)\rangle \\ Y_3 & \mapsto & T \\ Y_2 & \mapsto & Y \\ Y_1 & \mapsto & Z. \end{array}$$

Renaming (T, Z, Y) as (Y_3, Y_1, Y_2), we see that our result is

$$\mathbf{T}' \left| \begin{array}{l} K(Y_3, Y_1, Y_2) = Y_2^2 + Y_3^2 \\ H(Y_3, Y_1) = Y_1 + 100Y_3^2 \\ F(Y_3) = Y_3^4 + 1. \end{array} \right.$$

6.2 The bivariate case

The former example shows the importance of bivariate change of order. In this subsection, we give the details of the actual operation we need. We start from a univariate polynomial P in $\mathbb{F}_q[Y]$, and $A, V \in \mathbb{F}_q[Y]$, both reduced modulo P; our question is to determine whether there exists an isomorphism of the form

$$\begin{array}{ccc} \mathbb{F}_q[Y]/\langle P\rangle & \to & \mathbb{F}_q[Z,Y]/\langle Q(Z), S(Z,Y)\rangle \\ A & \mapsto & Z \\ V & \mapsto & Y, \end{array}$$

where $(Q(Z), S(Z, Y))$ is a bivariate triangular set in $\mathbb{F}_q[Z, Y]$. We will say that hypothesis (**h**) holds if such a triangular set exists.

Our purpose is to decide whether (**h**) holds, and if so, to compute $(Q(Z), S(Z, Y))$. To help, we will additionally impose the relation $Y = V + B(A) \bmod P$, for some given polynomial B in $\mathbb{F}_q[Z]$. Then, (**h**) holds if and only if there exists an isomorphism of the form

$$\begin{array}{ccc} \mathbb{F}_q[Y]/\langle P\rangle & \to & \mathbb{F}_q[Z,Y]/\langle Q(Z), R(Z,Y)\rangle \\ A & \mapsto & Z \\ Y & \mapsto & Y, \end{array}$$

the polynomials R and S being related by $S(Z, Y) = R(Z, Y + B) \bmod Q$. Equivalently, (**h**) holds if and only if the ideal $\langle P(Y), Z - A(Y)\rangle$ is generated by a triangular set of the form $(Q(Z), R(Z, Y))$. The following lemma shows how to use these remarks to solve our question; we rely on trace computations, where all traces are computed modulo P.

Lemma 16. *Let $d = \deg(P)$, and suppose that $q = p^n$, with p prime and $p > d$, and that P is squarefree. Given B such that $Y = V + B(A) \bmod P$, one can do the following using $d\,\mathrm{plog}(d)$ operations in $\mathbb{F}_q$:*

- *given $(\mathrm{tr}(A^j))_{j<d}$, verify whether (**h**) holds, and if so compute Q,*
- *given Q and $(\mathrm{tr}(Y^i A^j))_{i<d_1, j<d_2}$, with $d_2 = \deg(Q)$ and $d_1 = d/d_2$, compute S.*

Proof. The proof is a consequence of Lemma 12. In view of the remarks above, given $(\mathrm{tr}(A^j))_{j<d}$, we can use that lemma to verify whether (**h**) holds (and compute Q) using $d\,\mathrm{plog}(d)$ operations in $\mathbb{F}_q$. If the condition holds, from the traces $(\mathrm{tr}(Y^i A^j))_{i<d_1, j<d_2}$, we deduce R in a similar amount of time. Finally, S is deduced by a polynomial shift, with coefficients taken modulo Q. Using the divide-and-conquer algorithm of [17], this takes time $d\,\mathrm{plog}(d)$ as well. □

The following equivalent form of assumption (**h**) will be useful in the next subsection. The proof is similar to that of Lemma 10.

Lemma 17. *Let S_A be the $\mathbb{F}_q$-algebra generated by A in $\mathbb{F}_q[Y]/\langle P\rangle$. Then assumption (**h**) holds if and only if $\mathbb{F}_q[Y]/\langle P\rangle$ is a free S_A-module, generated by powers of V of the form $1, V, \ldots, V^e$.*

Proof. Since $Y = V + B(A)$, the second condition is equivalent to $\mathbb{F}_q[Y]/\langle P\rangle$ being a free S_A-module, generated by powers of Y of the form $1, Y, \ldots, Y^e$. We use this latter condition in the rest of the proof.

Let I be the ideal $\langle P(Y), Z - A(Y)\rangle$, so that $\mathbb{F}_q[Y]/\langle P\rangle$ is isomorphic to $\mathbb{F}_q[Z,Y]/I$. Then, S_A is isomorphic to the subalgebra $\mathbb{F}_q[Z]/Q$ of $\mathbb{F}_q[Z,Y]/I$, where Q is the minimal polynomial of A modulo P.

If (**h**) holds, then $\mathbb{F}_q[Y]/\langle P\rangle$ is isomorphic to $\mathbb{F}_q[Z,Y]/\langle Q(Z), R(Z,Y)\rangle$, and the conclusion follows.

Conversely, suppose that $\mathbb{F}_q[Y]/\langle P\rangle$ is a free S_A-module, generated by powers of Y of the form $1, Y, \ldots, Y^e$. Let $R(Z,Y)$ be the characteristic polynomial of Y in this free S_A-module, so that $\deg(R,Y) = e+1$; to conclude, we prove that $(Q(Z), R(Z,Y))$ generates I. Obviously, both Q and R are in I. Conversely, take F in I and let F' be its remainder modulo $\langle Q(Z), R(Z,Y)\rangle$. Then, $\deg(F',Y) \le e$, so we can write $F' = \sum_{i\le e} f_i(Z)Y^i$. Since F' is in I, all f_i must be zero modulo Q, that is, identically zero, and F' itself must be zero. □

6.3 The general case

Suppose now that s is arbitrary, let $\mathbf{Y} = Y_1, \ldots, Y_s$, and let I be a zero-dimensional radical ideal in $\mathbb{F}_q[\mathbf{Y}]$. We let R be the residue class ring $\mathbb{F}_q[\mathbf{Y}]/I$, and let δ be the dimension of R over $\mathbb{F}_q$. In all that follows, p denotes the characteristic of $\mathbb{F}_q$.

Our goal here is to decide whether there exists a triangular set $\mathbf{T}$ for the order $Y_1 < \cdots < Y_s$ such that $\langle \mathbf{T}\rangle = I$, and if so compute it. Our input is a univariate representation $\mathscr{P} = (Q, \boldsymbol{\lambda}, \mathbf{W})$ of I, with $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_s)$ and $\lambda_s = 1$.

Starting from $\mathscr{P}$, we will reintroduce the variables $Y_s, \dots, Y_1$ one by one, in this order, and eventually deduce $\mathbf{T}$ (or prove there is no such $\mathbf{T}$). Remark that $\mathscr{P}$ can be seen as a mixed representation $\mathscr{M}_s$ of format $(s, 1)$ for I. We will use it as the starting point for an iterative process, constructing mixed representations $\mathscr{M}_j$ of formats $(j, s-j+1)$, for $j = s-1, \dots, 1$.

We will thus say that $(\mathbf{H}_j)$ holds if I admits a mixed representation of format $(j, s-j+1)$. Then, we have the following:

- By the former remark, $(\mathbf{H}_s)$ holds.
- $(\mathbf{H}_1)$ holds if and only the triangular set $\mathbf{T}$ we are looking for exists; in this case, writing $\mathscr{M}_1 = (\mathbf{P}_1, \mathbf{V}_1, \boldsymbol{\ell}_1)$, and assuming $\boldsymbol{\ell}_1 = (1)$, we have $\mathbf{T} = \mathbf{P}_1$ (up to renaming the variables).
- Assuming $q \geq \delta^2$, if $(\mathbf{H}_{j-1})$ holds, then $(\mathbf{H}_j)$ holds (this is a consequence of Lemma 10).

Thus, starting from s, it is sufficient to iteratively test whether $(\mathbf{H}_{s-1}), \dots, (\mathbf{H}_1)$ hold, and if so compute corresponding mixed representations. If the test fails at any j, we know that $\mathbf{T}$ does not exist. The following lemma shows how to do the iterative step, from format $(j, s-j+1)$ to $(j-1, s-j+2)$.

Lemma 18. *Fix $\varepsilon > 0$, and suppose that the inequalities $q \geq \delta^2$ and $p > \delta$ hold. Given a mixed representation $\mathscr{M} = (\mathbf{P}, \mathbf{V}, \boldsymbol{\ell})$ of format $(j, s-j+1)$ for I, with $\boldsymbol{\ell} = (\ell_1, \dots, \ell_j)$ and $\ell_j = 1$, one can do the following using an expected $s\,\delta^{1+\varepsilon}\log(q)\operatorname{plog}_\varepsilon(\log(q))$ bit operations:*

- *decide whether $(\mathbf{H}_{j-1})$ holds;*
- *if so, compute a mixed representation $\mathscr{M}' = (\mathbf{P}', \mathbf{V}', \boldsymbol{\ell}')$ of format $(j-1, s-j+2)$ for I, with $\boldsymbol{\ell}' = (\ell'_1, \dots, \ell'_{j-1})$ and $\ell'_{j-1} = 1$.*

With notation as above, one can do the following using $\delta^{1+\varepsilon}\log(q)\operatorname{plog}_\varepsilon(\log(q))$ bit operations:

- *given $\mathscr{M}$, $\mathscr{M}'$ and A in $R_{\mathbf{P}}$, compute its image through the isomorphism $\Psi_{\mathscr{M}'} \circ \Phi_{\mathscr{M}} : R_{\mathbf{P}} \to R_{\mathbf{P}'}$.*
- *given $\mathscr{M}$, $\mathscr{M}'$ and A in $R_{\mathbf{P}'}$, compute its image through the isomorphism $\Psi_{\mathscr{M}} \circ \Phi_{\mathscr{M}'} : R_{\mathbf{P}'} \to R_{\mathbf{P}}$.*

Proof. Let $\mathscr{M} = (\mathbf{P}, \mathbf{V}, \boldsymbol{\ell})$ be a mixed representation of format $(j, s-j+1)$ for I, with $\mathbf{P} = (P, P_{j+1}, \dots, P_s)$ in $\mathbb{F}_q[Y, Y_{j+1}, \dots, Y_s]$, $\mathbf{V} = (V_1, \dots, V_j)$ in $\mathbb{F}_q[Y]$, $\boldsymbol{\ell} = (\ell_1, \dots, \ell_j)$, and $\ell_j = 1$.

Our first purpose is to find $\boldsymbol{\ell}' = (\ell'_1, \dots, \ell'_{j-1})$ in $\mathbb{F}_q^{j-1}$, with $\ell'_{j-1} = 1$ such that $\sum_{i \leq j-1} \ell'_i Y_i$ is a primitive element of level $j-1$. To do so, we choose $\boldsymbol{\ell}'$ at random, and test whether $Y_1, \dots, Y_{j-1}$ can be written as polynomials in $\sum_{i \leq j-1} \ell'_i Y_i$ modulo I. Recall that associated to $\mathscr{M}$, we have mutually inverse isomorphisms

$$\begin{array}{rccc} \Psi_{\mathscr{M}} : & R & \to & R_{\mathbf{P}} \\ & Y_1, \dots, Y_j & \mapsto & V_1, \dots, V_j \\ & Y_{j+1}, \dots, Y_s & \mapsto & Y_{j+1}, \dots, Y_s. \end{array} \quad \text{and} \quad \begin{array}{rccc} \Phi_{\mathscr{M}} : & R_{\mathbf{P}} & \to & R \\ & Y & \mapsto & \sum_{i \leq j} \ell_i Y_j \\ & Y_{j+1}, \dots, Y_s & \mapsto & Y_{j+1}, \dots, Y_s. \end{array}$$

Applying $\Psi_{\mathscr{M}}$, the former condition is equivalent to testing whether $V_1, \ldots, V_{j-1}$ can be written as polynomials in $A = \sum_{i \le j-1} \ell'_i V_i$ modulo P. This is done as follows:

- we compute the traces $\mathrm{tr}(A^k)_{k<2\delta}$ and $\mathrm{tr}(V_i A^k)_{k<\delta}$, for $i \le j-1$ (these are traces defined modulo P)
- using Lemma 11, we first compute the minimal polynomial Q of A modulo P, then candidates polynomials $V'_1, \ldots, V'_{j-1}$ in $\mathbb{F}_q[Z]$
- we test whether $V'_i(A) = V_i \bmod P$ for $i \le j-1$.

For a fixed $\boldsymbol{\ell}'$, in view of Lemma 11 and Theorem 3, going through this process takes an expected $s\,\delta^{1+\varepsilon}\log(q)\,\mathrm{plog}_\varepsilon(\log(q))$ bit operations. Since $q \ge \delta^2$, Lemma 9 shows that we expect to test 2 choices of $\boldsymbol{\ell}'$ before finding a primitive element of level $j-1$. Thus, the overall expected cost to find $\boldsymbol{\ell}'$ and $\mathbf{V}'$ is $s\,\delta^{1+\varepsilon}\log(q)\,\mathrm{plog}_\varepsilon(\log(q))$ bit operations.

Recall that $(\mathbf{H}_j)$ holds by assumption. Using the notation of Subsection 4.2, Lemma 10 implies that $(\mathbf{H}_{j-1})$ holds if and only if R_j is a free R_{j-1}-module, with a basis consisting of the first powers of Y_j. Through $\Psi_{\mathscr{M}}$, we have the isomorphism $R_j \simeq \mathbb{F}_q[Y]/\langle P\rangle$, R_{j-1} is the subring of R_j generated by A, and Y_j is mapped to V_j.

Let $B = \sum_{i \le j-1} \ell_i V'_i$, so that we have $Y = V_j + B(A) \bmod P$ (because $\ell_j = 1$). This remark allows us to apply Lemma 17: this shows that $(\mathbf{H}_{j-1})$ holds if and only if assumption $(\mathbf{h})$ of the last subsection holds, for the polynomials P, A and V_j. Using Lemma 16, we can thus decide whether $(\mathbf{H}_{j-1})$ holds, and if so compute polynomials $(Q(Z), S(Z,Y))$ that form a triangular set in $\mathbb{F}_q[Z,Y]$, and such that we have isomorphisms

$$\begin{array}{rccc} \psi: & \mathbb{F}_q[Y]/\langle P\rangle & \to & \mathbb{F}_q[Z,Y]/\langle Q,S\rangle \\ & Y & \mapsto & Y+B \end{array} \quad \text{and} \quad \begin{array}{rccc} \varphi: & \mathbb{F}_q[Z,Y]/\langle Q,S\rangle & \to & \mathbb{F}_q[Y]/\langle P\rangle \\ & Z & \mapsto & A \\ & Y & \mapsto & V_j. \end{array}$$

Remark in particular that ψ sends $V_1, \ldots, V_{j-1}, V_j$ to $V'_1, \ldots, V'_{j-1}, Y$. Computing all traces required by Lemma 16 and doing all post-processing fits into the same time bound as before.

Next, we reintroduce the variables $Y_{j+1}, \ldots, Y_s$. We will consider the triangular set $\mathbf{P}' = (Q, S, P'_{j+1}, \ldots, P'_s)$ in $\mathbb{F}_q[Z, Y, Y_{j+1}, \ldots, Y_s]$, with

$$P'_k = P_k(Y + B, Y_{j+1}, \ldots, Y_k) \bmod Q.$$

Recalling that we write $R_{\mathbf{P}} = \mathbb{F}_q[Y, Y_{j+1}, \ldots, Y_s]/\langle \mathbf{P}\rangle$ and $R_{\mathbf{P}'} = \mathbb{F}_q[Z, Y, Y_{j+1}, \ldots, Y_s]/\langle \mathbf{P}'\rangle$, we deduce the existence of the mutually inverse isomorphisms

$$\begin{array}{rccc} \Psi: & R_{\mathbf{P}} & \to & R_{\mathbf{P}'} \\ & Y & \mapsto & Y+B \\ & Y_{j+1}, \ldots, Y_s & \mapsto & Y_{j+1}, \ldots, Y_s \end{array} \quad \text{and} \quad \begin{array}{rccc} \Phi: & R_{\mathbf{P}'} & \to & R_{\mathbf{P}} \\ & Z & \mapsto & A \\ & Y & \mapsto & V_j \\ & Y_{j+1}, \ldots, Y_s & \mapsto & Y_{j+1}, \ldots, Y_s; \end{array}$$

as before, the former map sends $V_1, \ldots, V_{j-1}, V_j$ to $V'_1, \ldots, V'_{j-1}, Y$. Let us fix k and determine the cost of computing P'_k: this is done by applying ψ coefficient-wise, which amounts to $\deg(P_{j+1}, Y_{j+1}) \cdots \deg(P_k, Y_k)$ applications. By Theorem 3, each of them takes time

$d^{1+\varepsilon}\log(q)\,\mathrm{plog}_\varepsilon(\log(q))$, with $d = \deg(P)$. Since $d\deg(P_{j+1}, Y_{j+1})\cdots\deg(P_k, Y_k) = \delta$, all P'_k can be computed in an expected $s\,\delta^{1+\varepsilon}\log(q)\,\mathrm{plog}_\varepsilon(\log(q))$ bit operations.

Composing with $\Psi_{\mathscr{M}}$ and $\Phi_{\mathscr{M}}$ with Ψ and Φ, we obtain the following:

$$\begin{array}{ccc} R & \to & R_{\mathbf{P}'} \\ Y_1, \ldots, Y_{j-1} & \mapsto & V'_1, \ldots, V'_{j-1} \\ Y_j, \ldots, Y_s & \mapsto & Y, Y_{j+1}, \ldots, Y_s. \end{array} \quad \text{and} \quad \begin{array}{ccc} R_{\mathbf{P}'} & \to & R \\ Z & \mapsto & \sum_{i\le j-1} \ell'_i Y_i \\ Y, Y_{j+1}, \ldots, Y_s & \mapsto & Y_j, \ldots, Y_s. \end{array}$$

Thus, $\mathscr{M}' = (\mathbf{P}', \mathbf{V}', \boldsymbol{\ell}')$ is a mixed representation of format $(j-1, s-j+2)$ for I, and the previous maps are $\Psi_{\mathscr{M}'}$ and $\Phi_{\mathscr{M}'}$.

The final point to discuss is the cost of applying the isomorphisms $\Psi = \Psi_{\mathscr{M}'} \circ \Phi_{\mathscr{M}}$ and $\Phi = \Psi_{\mathscr{M}} \circ \Phi_{\mathscr{M}'}$. Both of them leave $Y_{j+1}, \ldots, Y_s$ unchanged, so these operations amount to apply $\deg(P_{j+1}, Y_{j+1})\cdots\deg(P_k, Y_k)$ times ψ and φ, respectively. By Theorem 3, each application takes time $d^{1+\varepsilon}\log(q)\,\mathrm{plog}_\varepsilon(\log(q))$, so the total time is $\delta^{1+\varepsilon}\log(q)\,\mathrm{plog}_\varepsilon(\log(q))$. □

As a consequence of the former lemma, we deduce the following result, which is the main point in this section. The proof is now straightforward.

Proposition 19. *Fix $\varepsilon > 0$. Then, given $\mathscr{P} = (Q, \boldsymbol{\lambda}, \mathbf{W})$ as above, such that $q \ge \delta^2$ and $p > \delta$, one can do the following in an expected $s^2\,\delta^{1+\varepsilon}\log(q)\,\mathrm{plog}_\varepsilon(\log(q))$ bit operations:*

- *decide whether there exists a triangular set $\mathbf{T}$ for the order $Y_1 < \cdots < Y_s$ that generates I*
- *if so, compute $\mathscr{M}_s, \ldots, \mathscr{M}_1$, with $\mathscr{M}_i = (\mathbf{P}_i, \boldsymbol{\ell}_i, \mathbf{V}_i)$, such that*
  - *$\mathscr{M}_s = \mathscr{P}$,*
  - *for $i = s-1, \ldots, 1$, $\mathscr{M}_i$ is obtained from $\mathscr{M}_{i+1}$ by means of Lemma 18,*
  - *$\mathbf{T} = \mathbf{P}_1$.*

With notation as before, one can do the following in $s\,\delta^{1+\varepsilon}\log(q)\,\mathrm{plog}_\varepsilon(\log(q))$ bit operations:

- *given $\mathscr{M}_1, \ldots, \mathscr{M}_s$ and A in $\mathbb{F}_q[Y]/Q$, compute its image in $R_{\mathbf{T}}$*
- *given $\mathscr{M}_1, \ldots, \mathscr{M}_s$ and A in $R_{\mathbf{T}}$ compute its image in $\mathbb{F}_q[Y]/Q$.*

6.4 Proof of Theorem 2

Finally, we prove Theorem 2. On input, we are given a triangular set $\mathbf{T}$ for the order $Y_1 < \cdots < Y_s$, with multidegree $\mathbf{d}$, and a target order $Y_{\sigma(1)} < \cdots < Y_{\sigma(s)}$ on the variables. We assume that the characteristic p of the base field satisfies $p > \delta_{\mathbf{d}}$.

We want to decide whether the ideal $\langle \mathbf{T} \rangle$ is radical, and if so, whether there exists a triangular set $\mathbf{T}'$ for the order $Y_{\sigma(1)} < \cdots < Y_{\sigma(s)}$ that generates the same ideal as $\mathbf{T}$. We also wish to do the change of bases $R_{\mathbf{T}} \to R_{\mathbf{T}'}$ and back.

First, if needed, we extend the base field, to ensure that the assumption $q \ge \delta_{\mathbf{d}}^2$ holds. This is done as in Subsection 5.4; since all the costs incurred by this field extension will fit in our target time complexity, for simplicity, we will still denote our base field by $\mathbb{F}_q$.

Applying Proposition 15, we can decide if $\mathbf{T}$ is squarefree, and if so, compute a primitive representation $\mathscr{P} = (P, \mathbf{V}, \boldsymbol{\ell})$, with $\ell_{\sigma(s)} = 1$, giving us inverse isomorphisms

$$\Psi_{\mathscr{P}} : \ R_{\mathbf{T}} \to \mathbb{F}_q[Y]/\langle P \rangle \quad \text{and} \quad \Phi_{\mathscr{P}} : \ \mathbb{F}_q[Y]/\langle P \rangle \to R_{\mathbf{T}}.$$

Let further $W_1 = V_{\sigma(1)}, \ldots, W_s = V_{\sigma(s)}$ be the images of $Y_{\sigma(1)}, \ldots, Y_{\sigma(s)}$ through $\Psi_{\mathscr{P}}$, and let $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_s)$, with $\lambda_i = \ell_{\sigma(i)}$. Finally, let $Z_1, \ldots, Z_s = Y_{\sigma(1)}, \ldots, Y_{\sigma(s)}$. As a result, $\mathscr{Q} = (P, \mathbf{W}, \boldsymbol{\lambda})$ is a primitive representation for the ideal $I = \langle \mathbf{T} \rangle$ in $\mathbb{F}_q[\mathbf{Z}]$, with $\lambda_s = 1$.

We are thus in a position to apply Proposition 19: this provides us with the triangular set $\mathbf{T}'$ (or proves it does not exist). The total time reported in Propositions 15 and 19 is an expected $s^2\, \delta^{1+\varepsilon} \log(q)\, \mathrm{plog}_\varepsilon(\log(q))$ bit operations.

To do the change of basis $R_{\mathbf{T}} \to R_{\mathbf{T}'}$, we first convert from $R_{\mathbf{T}}$ to $\mathbb{F}_q[Y]/\langle P \rangle$ using Proposition 15; this takes $s\, \delta^{1+\varepsilon} \log(q)\, \mathrm{plog}_\varepsilon(\log(q))$ bit operations. Then, mapping $\mathbb{F}_q[Y]/\langle P \rangle$ to $R_{\mathbf{T}'}$ is done by means of the second part of Proposition 19, for a similar amount of time. The inverse change of basis $R_{\mathbf{T}'} \to R_{\mathbf{T}}$ is done by first converting from $R_{\mathbf{T}'}$ to $\mathbb{F}_q[Y]/\langle P \rangle$, then to $R_{\mathbf{T}}$, using again Propositions 15 and 19. The time estimate is the same.

7 An illustration from elliptic curve point counting

In this section, we describe a situation similar to the example given in the introduction, where change of order was used to simplify factorization.

The following construction originates from point-counting algorithms for elliptic curves over finite fields. The objective is to count the number of points of an elliptic curve $E : Y^2 = X^3 + AX + B$ over $\mathbb{F}_p$; this is a fundamental operation in elliptic curve cryptology, see for instance [6]. In large characteristic, the best algorithms are based on Schoof's landmark contribution [39] and its improvements by Elkies [16] and Atkin [4].

These algorithms operate by Chinese Remaindering, by determining $|E|$ modulo various primes ℓ. For a given ℓ, Schoof's algorithm finds $|E| \bmod \ell$ by doing a search modulo the *division polynomial* ψ_ℓ. This polynomial has degree $(\ell^2 - 1)/2$; its roots are the (pairwise distinct) X-coordinates of the ℓ-torsion points on $|E|$.

Elkies proposed to improve this phase, by working only modulo a factor f_ℓ of ψ_ℓ of degree $(\ell - 1)/2$. This factor is obtained as follows:

- let $\Phi_\ell \in \mathbb{Z}[J, J']$ be the ℓth *modular polynomial* and let $\varphi_\ell = \Phi_\ell \bmod p$;
- compute a root α of $\varphi_\ell(J, j(E))$, where $j(E)$ is the j-invariant of E (or determine that no such root exists);
- if such a root exists, deduce f_ℓ from α.

We will not give more details here. It is enough to note that Φ_ℓ is a bivariate polynomial of degree ℓ^2, with coefficients of bit-size about ℓ, so the cost of Elkies' construction is $\Omega(\ell^3)$ bit operations. The primes ℓ for which the root α exists are called *Elkies primes*; conjecturally, for a given E, about half of the primes are Elkies primes. As a consequence, one may make the assumption that $\ell \leq \log(p)$; then, the cost for a given ℓ is an expected $\ell \log(p)^2$ bit operations, up to logarithmic factors [30].

We will discuss here an alternative to Elkies' construction, due to Charlap, Coley and Robbins [13]. A cost analysis is given in [36], with a result of the form $\ell^4 + \ell \log(p)$ operations in $\mathbb{F}_p$ for a given ℓ, which is $\ell^4 \log(p) + \ell \log(p)^2$ bit operations (omitting logarithmic factors in both cases). We will show that our results allow us to reduce the cost and make it comparable to the one of Elkies' algorithm.

Again, the purpose is to find a suitable factor f_ℓ of ψ_ℓ; now, this is done by purely "algebraic" means, whereas Elkies' approach relies on transcendental arguments. Let $[m]$ denote the multiplication-by-m map on E; then, there exist rational functions γ_m, η_m in $\mathbb{F}_p(X)$ such that for all (x, y) in E, we have

$$[m](x, y) = (\gamma_m(x), y\eta_m(x));$$

γ_m has a pole at x if and only if $[m](x, y)$ is a zero on E. Let $R = \mathbb{F}_p[X]/\psi_\ell$, and for $m < \ell$, let $g_m \in R$ be the image of γ_m in R; $\gamma_m \bmod \psi_\ell$ is well-defined, as one easily sees that its denominator has no common root with ψ_ℓ. Finally, define $A = \sum_{i=1}^{(\ell-1)/2} g_i$.

Lemma 20. *Let m_A be the minimal polynomial of A modulo ψ_ℓ.*

- *m_A has degree at most $\ell + 1$.*
- *if $\deg(m_A) = \ell + 1$, then $\chi_A = m_A^{(\ell-1)/2}$.*

Proof. Let x be a root of ψ_ℓ, and let $P = (x, y)$ be a corresponding ℓ-torsion point on E. Then, $A(x)$ is the sum of the abscissas of the points $P, \ldots, [(\ell - 1)/2]P$. In particular, $A(x) = A(g_2(x)) = \cdots = A(g_{(\ell-1)/2}(x))$; thus, the roots of ψ_ℓ can be partitioned into $\ell + 1$ subsets over each of which A takes a constant value. The conclusion follows from Equation (1) of Section 4. □

Let us suppose that we are in the case where $\deg(m_A) = \ell + 1$ (this is true "in general", see [13]). Then, the former lemma implies that the ideal $\langle \psi_\ell(Y), Z - A(Y) \rangle$ is generated by the triangular triangular set $(P(Z), Q(Z, Y))$, with $P = m_A$, $\deg(P) = \ell+1$ and $\deg(Q, Y) = (\ell - 1)/2$. Furthermore, Charlap, Coley and Robbins prove that ℓ is an Elkies prime if and only if P has a root α in $\mathbb{F}_p$; in this case, $Q(\alpha, Y)$ is the factor f_ℓ we are looking for.

The following theorem gives a cost estimate on the computation of this factor, which shows it to be roughly as costly as Elkies' approach: if one could take $\varepsilon = 0$ below, the second term would become dominant, and the overall cost would be $\ell \log(p)^2 \operatorname{plog}(\log(p))$, as for Elkies' algorithm.

Theorem 21. *Fix $\varepsilon > 0$. If $\ell \leq \log(p)$, one can compute an Elkies factor (or determine that none exists) in an expected $\left(\ell^{2+\varepsilon} \log(p) + \ell \log(p)^2\right) \operatorname{plog}_\varepsilon(\ell \log(p))$ bit operations.*

Proof. First, we compute ψ_ℓ, by the standard binary powering scheme [6]; this can be done using $\ell^2 \log(p) \operatorname{plog}(\ell \log(p))$ bit operations.

Next, we show how to compute A for the cost of $O(\log(\ell))$ modular compositions modulo ψ_ℓ, using binary powering techniques inspired by the trace computation of [19] (see [32]

for a previous instance of the following computation). Let τ be a generator of $\mathbb{F}_\ell^*$; then, $A = \sum_{i=0}^{(\ell-1)/2-1} g_{\tau^i}$. For $g \geq 0$, let

$$G_k = g_{\tau^{2^k}} \quad \text{and} \quad A_k = \sum_{i=0}^{2^k-1} g_{\tau^i}.$$

For simplicity, we show here how to compute A_k. Since we have $g_a(g_b) \bmod \psi_\ell = g_{ab}$, it is sufficient to compute

- $G_0, G_1, \ldots, G_k$ by means of $G_{j+1} = G_j(G_j) \bmod \psi_\ell$
- $A_0, A_1, \ldots, A_k$ by means of $A_{j+1} = A_j(G_j) + A_j \bmod \psi_\ell$.

The slightly more general question of computing A (which involves summation bounds that are not powers of 2) is handled similarly. In any case, this requires $O(\log(\ell))$ modular compositions modulo ψ_ℓ; the cost is an expected $\ell^{2+\varepsilon}\log(p)\,\mathrm{plog}_\varepsilon(\log(p))$ bit operations, by Theorem 1.

Once A is known, we apply the change of order algorithm of Theorem 2 to the system $(\psi_\ell(Y), Z - A(Y))$; this is valid, since ψ_ℓ is squarefree, and $p \geq \ell^2$. This gives us the triangular set $(P(Z), Q(Z,Y))$ for the same expected $\ell^{2+\varepsilon}\log(p)\,\mathrm{plog}_\varepsilon(\log(p))$ bit operations. Finally, we find a root α of the minimal polynomial P (if any) in an expected $\ell\log(p)^2\,\mathrm{plog}(\ell\log(p))$ bit operations [18, Corollary 14.16]. We deduce the factor $Q(\alpha, Y)$ by evaluation, for a cost linear in $\ell^2\log(p)$, up to logarithmic factors. □

Acknowledgments

Adrien Poteaux was supported by the EXACTA grant of the National Science Foundation of China (NSFC 60911130369), the French National Research Agency (ANR-09-BLAN-0371-01) and the European union (PITN-GA-2008-214584 SAGA). Éric Schost is supported by NSERC and the Canada Research Chair program.

References

[1] C. J. Accettella, G. M. Del Corso, and G. Manzini. Inversion of two level circulant matrices over Zp. *Linear Algebra and its Applications*, 366:5 – 23, 2003.

[2] Alfred V. Aho, John E. Hopcroft, and Jeffrey D. Ullman. *The Design and Analysis of Computer Algorithms*. Addison-Wesley, 1974.

[3] M. E. Alonso, E. Becker, M.-F. Roy, and T. Wörmann. Zeros, multiplicities and idempotents for zerodimensional systems. In *MEGA 94*, volume 142 of *Progress in Mathematics*, pages 1–15. Birkhäuser, 1996.

[4] A. O. L. Atkin. The number of points on an elliptic curve modulo a prime (II). Available at `http://listserv.nodak.edu/archives/nmbrthry.html`, 1992.

[5] P. Aubry, D. Lazard, and M. Moreno Maza. On the theories of triangular sets. *Journal of Symbolic Computation*, 28(1, 2):45–124, 1999.

[6] I. Blake, G. Seroussi, and N. Smart. *Elliptic curves in cryptography*, volume 265 of *London Mathematical Society Lecture Notes Series*. Cambridge University Press, 1999.

[7] A. Bostan, P. Flajolet, B. Salvy, and É. Schost. Fast computation of special resultants. *Journal of Symbolic Computation*, 41(1):1–29, 2006.

[8] A. Bostan, G. Lecerf, and É. Schost. Tellegen's principle into practice. In *ISSAC'03*, pages 37–44. ACM, 2003.

[9] A. Bostan, M. F. I. Chowdhury, J. van der Hoeven, and É. Schost. Homotopy methods for multiplication modulo triangular sets. *Journal of Symbolic Computation*. To appear.

[10] F. Boulier, F. Lemaire, and M. Moreno Maza. PARDI! In *ISSAC'01*, pages 38–47. ACM, 2001.

[11] R. P. Brent and H. T. Kung. Fast algorithms for manipulating formal power series. *Journal of the ACM*, 25(4):581–595, 1978.

[12] P. Bürgisser, M. Clausen, and M. A. Shokrollahi. *Algebraic Complexity Theory*. Springer, 1997.

[13] L. S. Charlap, R. Coley, and D. P. Robbins. Enumeration of rational points on elliptic curves over finite fields. Draft, 1991.

[14] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. *Journal of Symbolic Computation*, 9(3):251–280, 1990.

[15] X. Dahan, M. Moreno Maza, É. Schost, and Y. Xie. On the complexity of the D5 principle. In *Transgressive Computing*, pages 149–168, 2006.

[16] N. Elkies. Explicit isogenies. Draft, 1992.

[17] J. von zur Gathen. Functional decomposition of polynomials: the tame case. *Journal of Symbolic Computation*, 9:281–299, 1990.

[18] J. von zur Gathen and J. Gerhard. *Modern Computer Algebra*. Cambridge University Press, 1999.

[19] J. von zur Gathen and V. Shoup. Computing Frobenius maps and factoring polynomials. *Computational Complexity*, 2(3):187–224, 1992.

[20] P. Gianni and T. Mora. Algebraic solution of systems of polynomial equations using Gröbner bases. In *AAECC'5*, volume 356 of *Lecture Notes in Computer Science*, pages 247–257. Springer Verlag, 1989.

[21] M. Giusti, J. Heintz, J. E. Morais, J. Morgenstern, and L. M. Pardo. Straight-line programs in geometric elimination theory. *Journal of Pure and Applied Algebra*, 124:101–146, 1998.

[22] M. Giusti, G. Lecerf, and B. Salvy. A Gröbner free alternative for polynomial system solving. *Journal of Complexity*, 17(1):154–211, 2001.

[23] X. Huang and V. Y. Pan. Fast rectangular matrix multiplication and applications. *Journal of Complexity*, 14(2):257–299, Jun 1998.

[24] É. Hubert. Notes on triangular sets and triangulation-decomposition algorithms. I. Polynomial systems. In *Symbolic and numerical scientific computation*, volume 2630 of *Lecture Notes in Computer Science*, pages 1–39. Springer, 2003.

[25] M. Kalkbrener. A generalized euclidean algorithm for computing triangular representations of algebraic varieties. *Journal of Symbolic Computation*, 15:143–167, 1993.

[26] E. Kaltofen. Greatest common divisors of polynomials given by straight-line programs. *Journal of the ACM*, 35(1):231–264, 1988.

[27] E. Kaltofen. Challenges of symbolic computation: my favorite open problems. *Journal of Symbolic Computation*, 29(6):891–919, 2000.

[28] E. Kaltofen and Y. Laskhman. Improved sparse multivariate polynomial interpolation algorithms. In *ISSAC'88*, volume 358 of *Lecture Notes in Computer Science*, pages 467–474. Springer Verlag, 1989.

[29] K. S. Kedlaya and C. Umans. Fast polynomial factorization and modular composition. To appear, available at `http://www.cs.caltech.edu/~umans/papers/KU08-final.pdf`.

[30] R. Lercier and T. Sirvent. Elkies subgroups of elliptic curve ℓ-torsion points. *Journal de Théorie des Nombres de Bordeaux*, 20(3):783–797, 2008.

[31] X. Li, M. Moreno Maza, and É. Schost. Fast arithmetic for triangular sets: from theory to practice. *Journal of Symbolic Computation*, 44(7):891–907, 2009.

[32] F. Morain, P. Mihailescu, and É. Schost. Computing the eigenvalue in the Schoof-Elkies-Atkin algorithm using Abelian lifts. In *ISSAC'07*, pages 285–292. ACM, 2007.

[33] M. Moreno Maza. On triangular decompositions of algebraic varieties. Technical Report TR 4/99, NAG Ltd, Oxford, UK, 1999. `http://www.csd.uwo.ca/~moreno/`.

[34] V. Y. Pan. Simple multivariate polynomial multiplication. *Journal of Symbolic Computation*, 18(3):183–186, 1994.

[35] C. Pascal and É. Schost. Change of order for bivariate triangular sets. In *ISSAC'06*, pages 277–284. ACM, 2006.

[36] C. Peters. Bestimmung des Elkies-Faktors im Schoof-Elkies-Atkin-Algorithmus, 2006. Diploma Thesis, Universität Paderborn.

[37] Daniel Reischert. Asymptotically fast computation of subresultants. In *Proceedings of the 1997 international symposium on Symbolic and algebraic computation*, ISSAC '97, pages 233–240, New York, NY, USA, 1997. ACM.

[38] F. Rouillier. Solving zero-dimensional systems through the Rational Univariate Representation. *Applicable Algebra in Engineering, Communication and Computing*, 9(5):433–461, 1999.

[39] R. Schoof. Elliptic curves over finite fields and the computation of square roots mod p. *Mathematics of Computation*, 44:483–494, 1985.

[40] É. Schost. Complexity results for triangular sets. *Journal of Symbolic Computation*, 36(3–4):555–594, 2003.

[41] V. Shoup. New algorithms for finding irreducible polynomials over finite fields. *Mathematics of Computation*, 54(189):435–447, 1990.

[42] V. Shoup. A fast deterministic algorithm for factoring polynomials over finite fields of small characteristic. In *ISSAC'91*, pages 14–21. ACM, 1991.

[43] V. Shoup. Fast construction of irreducible polynomials over finite fields. *Journal of Symbolic Computation*, 17(5):371–391, 1994.

[44] C. Umans. Fast polynomial factorization and modular composition in small characteristic. In *STOC*, pages 481–490, 2008.

[45] L. Yang, X. Hou, and B. Xia. A complete algorithm for automated discovering of a class of inequality-type theorems. *Science in China. Series F. Information Sciences*, 44(1):33–49, 2001.